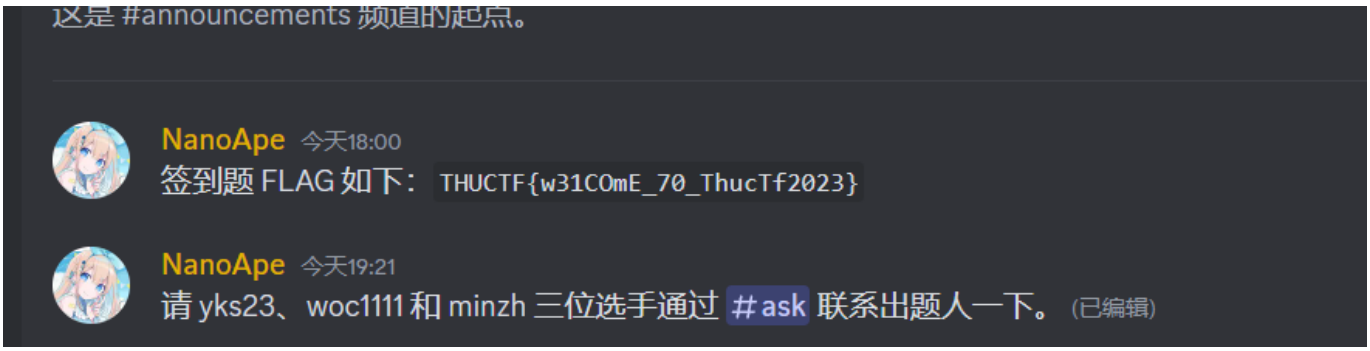
THUCTF2023

这是一份THUCTF的Writeup By Sceleri。

用markdown写的，所以pdf的排版可能会比较抽象。

一道难题

base64解密即可



呀哈哈

图片隐写题，把所有方法全试一遍就行（？）

最后发现修改长宽后还有内容，拿tweakpng改了就行。

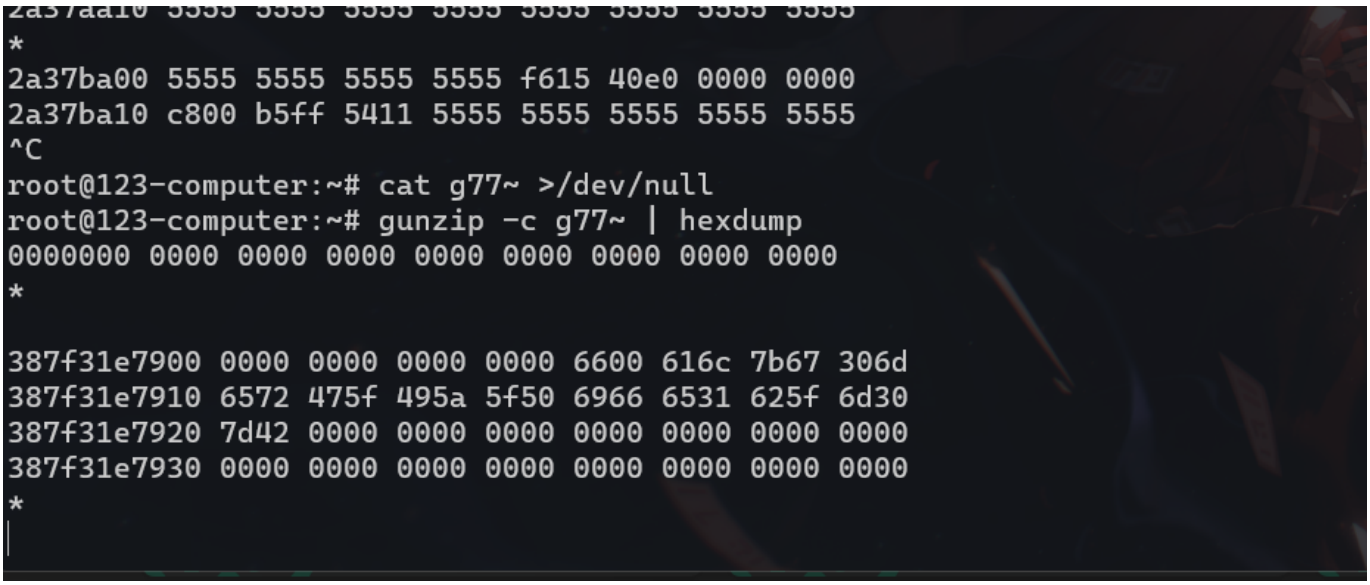
KFC

没有一点地名（看不清），不得已谷歌识图然后就出来了。

截图忘截了，反正是巴黎的一家店。

未来磁盘

直接解压！



Dark(er) Room

翻看源码可以获得地图以及各种事件信息（但其实都是靠试）

然后出来后知道需要117%的sanity，于是优化了路线，最后只需要连过3次help即可，算算概率发现竟然有1/125，直接写脚本刷。

在地图里可以发现一个flagroom，尝试获取flag可以发现它要我们猜公钥，于是乱输一堆搞崩后发现它在flag_number末位为1时会更新公钥且更新一次需要花1s，于是写一个脚本记一下时即可。

两道题都是拿pwntools写的。

```python
from pwn import *
import time

io = process("nc chal.thuctf.redbud.info 50825",shell=True)
def lazysend(s):
    io.sendline(s)
    time.sleep(0.05)

def flag1():
    global io
    io = process("nc chal.thuctf.redbud.info 50825",shell=True)
    lazysend("newgame")
    lazysend("123")
    lazysend("y")
    lazysend("n")
    lazysend("n")
    lazysend("e")
    lazysend("pickup key")
    lazysend("w")
    lazysend("s")
    lazysend("s")
    lazysend("e")
    lazysend("e")
    lazysend("e")
    lazysend("pickup trinket")
    lazysend("w")
    lazysend("s")
    lazysend("usewith key door")
    lazysend("s")
    lazysend("s")
    lazysend("n")
    lazysend("w")
    lazysend("w")
    lazysend("w")
    lazysend("n")
    lazysend("pickup key")
    lazysend("s")
    lazysend("e")
    lazysend("e")
    lazysend("e")
    lazysend("n")
    lazysend("n")
    lazysend("w")
    lazysend("use trinket")
    lazysend("w")
    lazysend("n")
    lazysend("n")
```

```
    lazysend("w")
    lazysend("w")
    lazysend("usewith key door")
    lazysend("h")
    lazysend("h")
    lazysend("h")
    lazysend("n")
    re = io.recvuntil(b"You have escaped with",timeout=1).decode('utf-8')
    if "THUCTF" in re:
        print(re)
    result = io.recvline().decode('utf-8').strip().replace("% sanity.",'')
    print(int(result))
    if int(result)>=110:
        io.interactive()
    io.close()

for i in range(200):
    flag1()

def flag2():
    lazysend("newgame")
    lazysend("123")
    lazysend("y")
    lazysend("n")
    lazysend("n")
    lazysend("n")
    lazysend("n")
    lazysend("w")
    lazysend("w")
    lazysend("s")
    lazysend("getflag")
    sss=''
    def getbyte():
        count = 0
        global sss
        for i in range(8):
            s=time.time()*1000
            io.sendline("0")
            io.recvuntil(b"Wrong",timeout=5)
            e=time.time()*1000
            if e-s>900:
                count+=2**i
            time.sleep(0.05)
        sss = chr(count)+sss
        print(chr(count),":",count,"    ",sss)

    while True:
        getbyte()
```

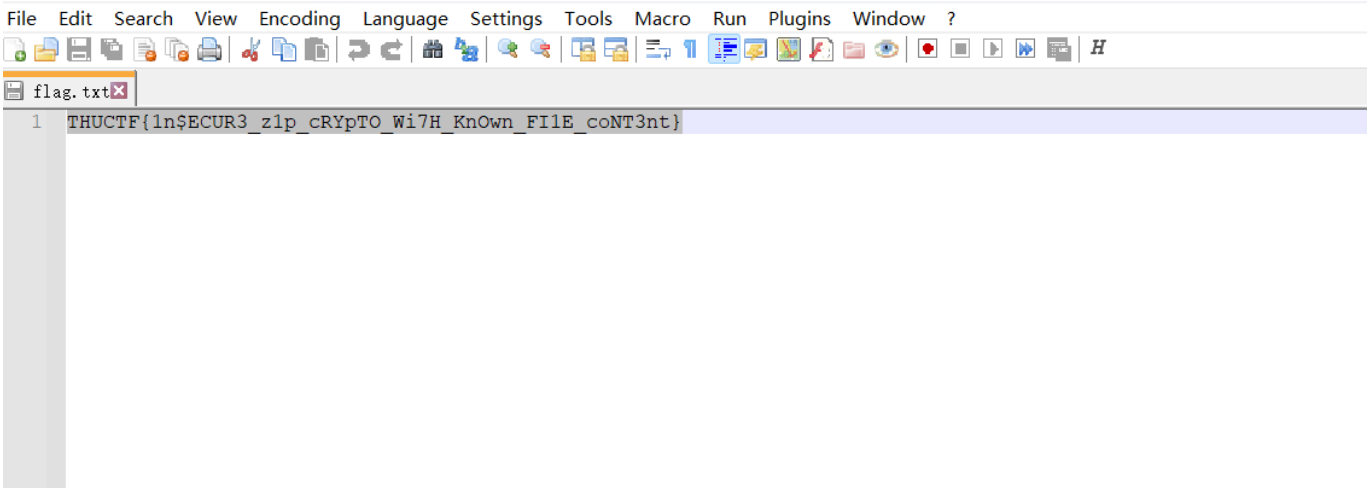
```
[*] Stopped process '/bin/sh' (pid 31868)
[+] Starting local process '/bin/sh': pid 31891
52
[*] Stopped process '/bin/sh' (pid 31891)
[+] Starting local process '/bin/sh': pid 31908
102
[*] Stopped process '/bin/sh' (pid 31908)
[+] Starting local process '/bin/sh': pid 31931
87
[*] Stopped process '/bin/sh' (pid 31931)
[+] Starting local process '/bin/sh': pid 31948
117
[*] Switching to interactive mode
You got flag1: THUCTF{fLa9I_yOu_PLAY_gaME_VeRY_W3ll}
BTW, if you finish with 117\% or higher sanity, you can get flag1

[...]: $
```

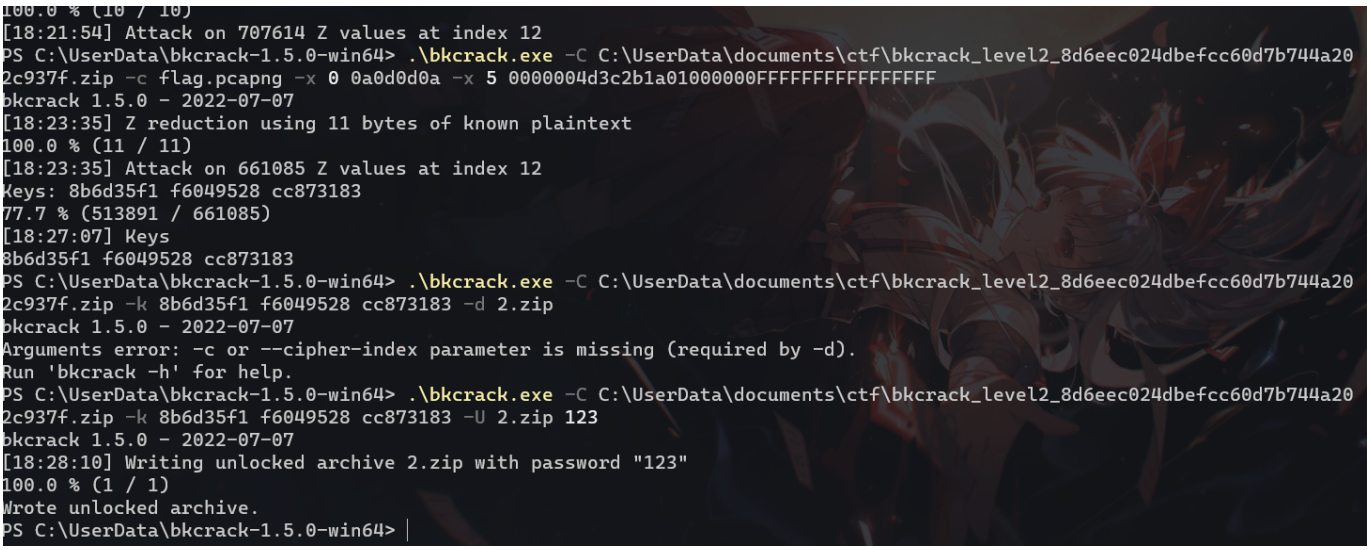
```python
lazysend("w")
lazysend("w")
lazysend("s")
lazysend("getflag")
sss=''
def getbyte() -> None:
    count = 0
    global sss
    for i in range(8):
        s=time.time()*1000
        io.sendline("0")
        io.recvuntil(b"Wrong",timeout=5)
        e=time.time()*1000
        if e-s>900:
            count+=2**i
        time.sleep(0.05)
    sss = chr(count)+sss
    print(chr(count),":",count,"   ",sss)

while True:
    getbyte()
#io.interactive()
```

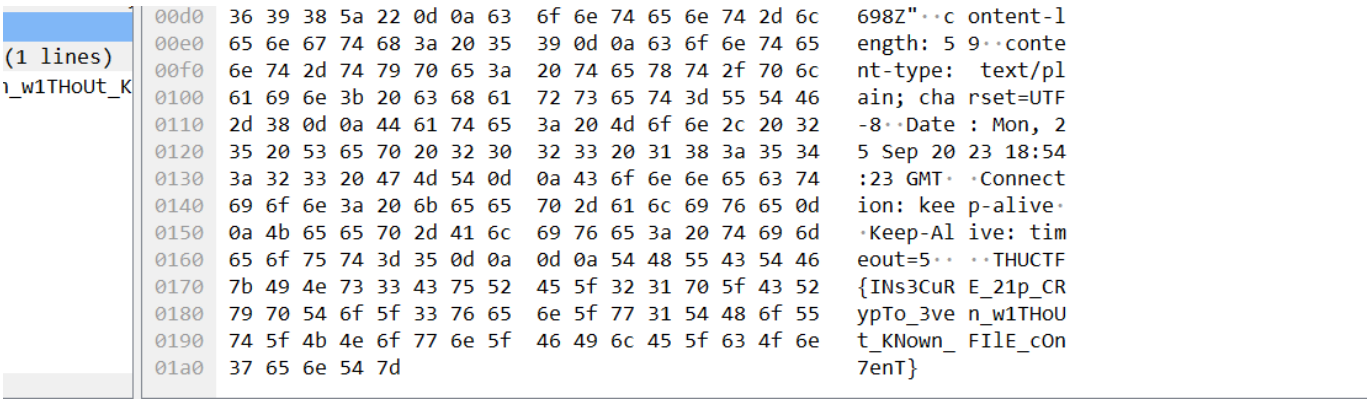
```
问题  输出  调试控制台  终端  端口  1

: 85    U_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 111    oU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 89    YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 95    _YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 50    2_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 54    62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 65    A62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 76    LA62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 49    1LA62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 123    {1LA62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 70    F{1LA62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 84    TF{1LA62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 67    CTF{1LA62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
: 85    UCTF{1LA62_YoU_s0lVeD_thIS_CH4lIENG3_ExcEI1ENT1y}
raceback (most recent call last):
File "/root/darkroom.py", line 35, in <module>
  getbyte()
File "/root/darkroom.py", line 26, in getbyte
  io.recvuntil(b"Wrong",timeout=5)
File "/usr/local/lib/python3.10/dist-packages/pwnlib/tubes/tube.py", line 341, in recvuntil
  res = self.recv(timeout=self.timeout)
File "/usr/local/lib/python3.10/dist-packages/pwnlib/tubes/tube.py", line 106, in recv
  return self._recv(numb, timeout) or b''
File "/usr/local/lib/python3.10/dist-packages/pwnlib/tubes/tube.py", line 176, in _recv
  if not self.buffer and not self._fillbuffer(timeout):
```

基本功

zip加密且密钥没法爆破，因此只能明文攻击。注意到文件名是一个zip爆破软件，所以直接找明文即可。第一题去找chromedriver，直接搜索文件大小就找得到。第二题是pcapng文件，在下面的题里用过，打开看看，再跟网上搜到的文件格式对比一下就可以猜出前16bytes的15个，然后用-x爆破即可。解出来后打开文件在最后发现了flag。

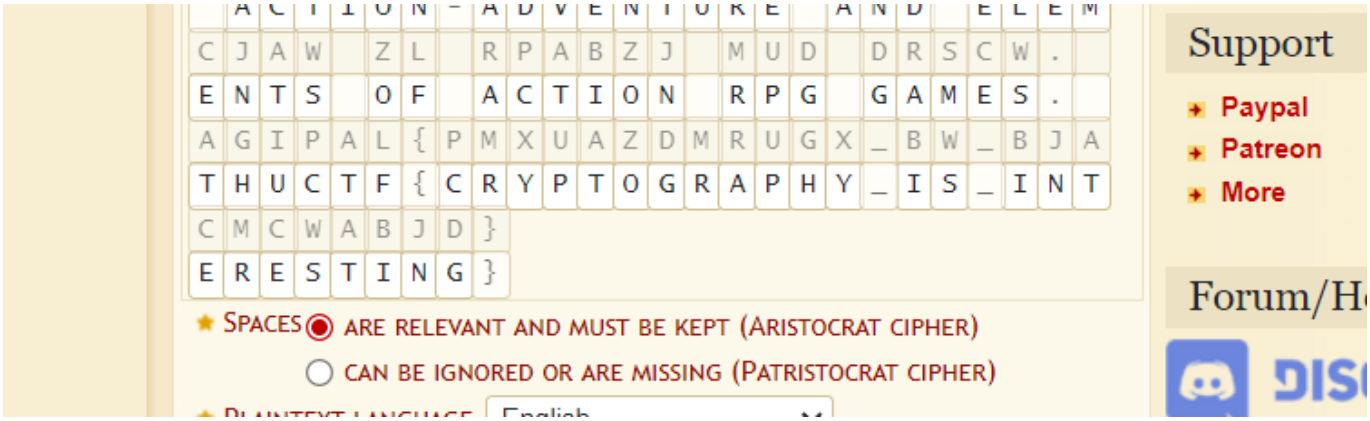


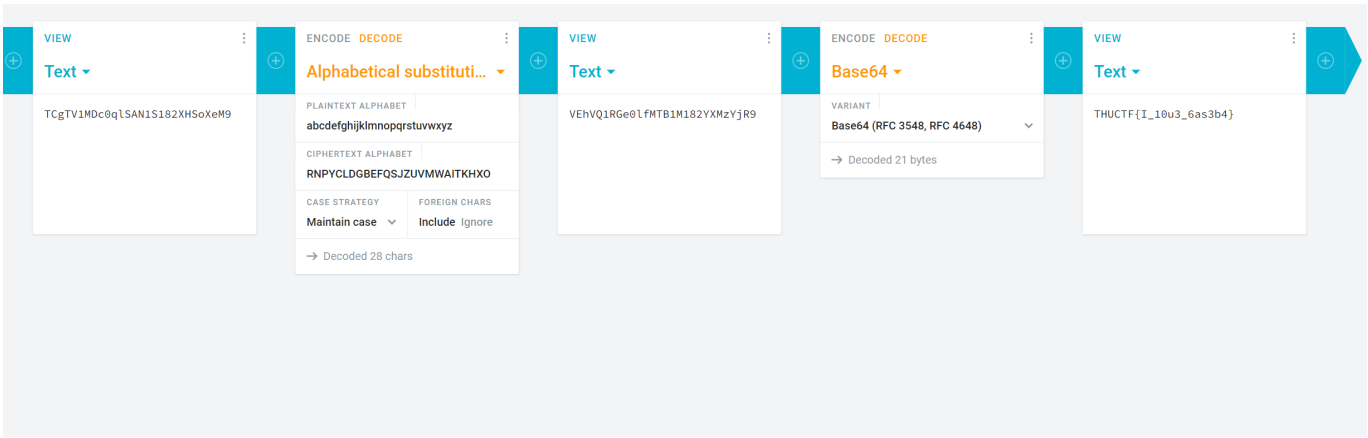




easycrypto

分析两个给的文件发现一个是字符替换另一是base64，且使用了同一套字符表，所以放在一起分析。随便在网上找了一个解密器，然后第一题就做出来了。然后去解第二题时发现字符表有很多没用过的字符有错，然后就开始痛苦修bug。。。反正靠ascii的字符编码比较稀疏强行修就行了（）





cookies

在网上搜索后发现random库在624个int32后会可以预测，所以直接从网上找了一个库然后把前2496位输进去就解决了第一题。第二题的entropy太小了，直接搜索即可。

```python
from random import Random
from randcrack import RandCrack



seed1 = 0x7119E49BD3EC88C863293AB8018E7BDE46242185BBEBE0E1FC5ED8563C605F86
seed2 = 0x0
message = "这里是密文"

message = bytes.fromhex(message)
# print(message)
print(len(message))

def xor_arrays(a, b, *args):
    if args:
        return xor_arrays(a, xor_arrays(b, *args))
    return bytes([x ^ y for x, y in zip(a, b)])

def guess(entropy):
    rc = RandCrack()
    void1 = Random(seed1)
    void2 = Random(seed2)
    void1.randbytes(entropy)
    void2.randbytes(entropy)
    ancient = xor_arrays(
        message, void1.randbytes(len(message)), void2.randbytes(len(message))
    )
    ancient = message
    for i in range(624):
        rc.submit(int.from_bytes(ancient[4 * i : 4 * i + 4],
byteorder="little"))

    cwedq = rc.predict_getrandbits(len(message) * 8 - 624 * 32).to_bytes(
        len(message) - 624 * 4, byteorder="little"
    )
    return xor_arrays(ancient[4 * 624 :], cwedq)


# i = 0
# while i < (2 << 22):
#     r = guess(i)
#     if b"THUCTF" in r:
#         print(i)
#         print(r)
print(guess(0))
```

```
(usual) C:\UserData\documents\ctf>C:/Users/123/.conda/envs/usual/python.exe c:/UserData/documents/ctf/cookiebreak_1.py
2544
b'\x00\x00\x00\x00THUCTF{ec7178c1-c9b1-4a05-a6d2-24a0a6a2dc0e}'

(usual) C:\UserData\documents\ctf>
```

```
0
b'\x00\x00\x00\x00THUCTF{a147b036-e62e-42c6-a46e-73866d2b0435}'
0
b'\x00\x00\x00\x00THUCTF{a147b036-e62e-42c6-a46e-73866d2b0435}'
0
b'\x00\x00\x00\x00THUCTF{a147b036-e62e-42c6-a46e-73866d2b0435}'
Traceback (most recent call last):
  File "c:\UserData\documents\ctf\cookiebreak_1.py", line 41, in <module>
    r = guess(i)
        ^^^^^^^^
  File "c:\UserData\documents\ctf\cookiebreak_1.py", line 31, in guess
    rc.submit(int.from_bytes(ancient[4 * i : 4 * i + 4], byteorder="little"))
  File "C:\Users\123\.conda\envs\usual\Lib\site-packages\randcrack\randcrack.py", line 16, in submit
    self.mt.append(self._harden_inverse(bits))
                   ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\123\.conda\envs\usual\Lib\site-packages\randcrack\randcrack.py", line 182, in _harden_inverse
    bits = self._decode_harden_midop(bits, self._to_bitarray(0x9d2c5680), 7)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\123\.conda\envs\usual\Lib\site-packages\randcrack\randcrack.py", line -1, in _decode_harden_midop
KeyboardInterrupt
```

第三题似乎只需要输入相同的seed即可通过。然后测试的时候发现少输几个不会报错，结果只输入了一个就过了。。。原来是zip只会依据长度小的那个来结束（我不好评价）

```
f937eafd982fe3cd,0x5f818300b266a17788f6900a2d3e4402d787f66b508262a310764d8ba6256fbe,0x95eef554674409f6eaf360b1bbf
5aa3b11ac9fb26d37305d915e9f8dc,0x62071c04330cfd5b9fce4ad4a4ec185e39de44c33703995dfd4077aa335dd019,0xda1095f891b7a
1d30d5d1d6879dfb8fe69a385e2262d0072e56196591,0xaa6bffd41404d785824070b10bd6937560d88e1342785d565ea2f8297e4fc030,0
fc9c65f372c5c66e81863e73fb7dcbb6b166b33814ad1dd703ebcdbd33,0xc8767cdf9622fcd2950b93d282e8ad4455b114047b83419e82d7
16af9,0xfc9f9c591302635822bdd1482362b7dacb739aea24f5c2e6d37d7d35797f9c3c,0xd367f0dc743a24a5a1aa8d472b2d295ba7281a
3cea1f7c9bdb432a7e6,0xf3eeffb4041a70d4fd9c846c5ebf27da81eb53c7171fe73913b0019e35fdbea3,0xf5f09b658703533f96ba5157
8b659f1507343318cd634366f48e0fcf2,0x91c86d11822711427e17d274b987aade878234f3e8051addf67b607689b9778f,0x1220c91880
42507aa206997dae05b2e4fc960e4fb3ff0dafd04ccf2a,0x3a78e88b6be1864be771678e28409c835a6fb09bfcdc229522790dbbd5de75f7
70095b498f1a9be7bd6c1943b9d97866aeb4b42cc17d7af4bef0c38afba9,0x9e3c9424ec8fb96726bc778a4f64a126b08a1185575e711b16
bd7f6a8,0xefa604fc36d58003399 44fd594d3e09e69417982ba482b382aecbf2ede86969d,0x2299c2d0690a6510e9c22aa16d0053686b73
aa624e5ad03c690341bf4,0xd543c3a653b620618ebfc3dcc86b97cbbd68f1d6b0e2a66acd812e9b1784326a,0xf8ea144a8a84a2590a5852
359461d315a57c19cf0a7adfc4b5b4773d2,0x2221beff77ddd51717de1ffa89064eff594afe8ee8add398085f815e05287aa9,0x1019ca07
fb26c5f6f42fed7067f80a7aeeb45a6f931fbaf2ba0bd494d,0xf063d7ce4973f897dd18b92fa7b8b4f3b95e66fc7f0b2a2cd78adb109ed5c
63fd3dae04b3994f6ffb7b2e43bf4e9323cac6f40a79ebca25cb7c8d200500b,0x9e90ae4820f615c966a271ee6e4305451f2df5a6d8d0bfa
e6ddb1892d,0x4d7b7c75d8a6bdac7411199d818e205470af8ec33d8da6d5fe2098ce38968425,0x55558f05319b806a8aff25c20f0ebb9c3
bf661d6c590406a6806e0fbc,0xff39795a74382ca938405f844bca8daacb0cccb6bfdafa5bf093cd6ad98fd66d,0xbb41652f6a1971de2ce
eda0b7c5aa0f9915b8f42fefe859cd0da01beb,0x57f895061943c1198aed19719da8fcf079f7a77aa03366de695b25bfa470bfac,0xd8a99
2970f1d0a09e3fb0aeeeb63fd043229d0712875cd05710d6500c,0x43b41ba0adf2b4b935bc188ba64f0fb7de774c8f2bc82b8c64e55fafeb
0x2a1ff09edf2cd0ee789277d05552dc3abbc298854c47ff3e1fc76131443a8b1d,0xb25351be1467901ea43dc15b0aa23fc0ec15ebcfd985
d65dcf59dc0b9,0xd46ae4c1d096d8ba51e6a5314b3e5fd19f3abbc4d1bc7c72b64654f04c3121ad,0xa2783f6ab122f044d4512c907d4c96
6e2bc3b1fff3460b60fb7e8086f,0xfa0737e9ede74c097a4f5ef63a4df50684417b547e92a3d828c3aa3216670c01>

wh@t r3sponse d0s t#ou pr0f3r, puny m0rt@l?
> 0xf6f38359827225cb2325a14ac49cf1606f8116d6f48e7f279b16ad78a7f4c0d8
Good job, Smol Tako! Here's your delicious SUPA BIG cookie! uwu
THUCTF{56d89afa-caf3-4806-a17a-5d3029f144d0}
```

Another V Me 50

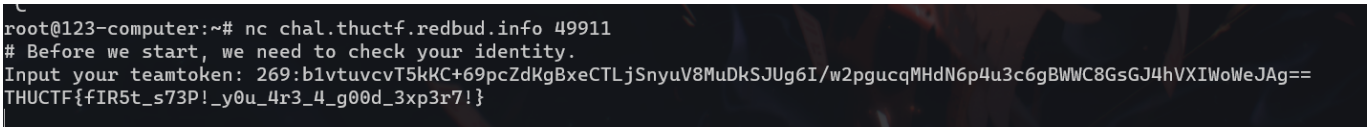
阅读源码可以发现一个账户要有50需要重复注册，而token只是sha256后14位，所以可以去撞哈希值。直接拿python的dict随便写了一个，然后它吃了16G内存。。。不过跑出来了就行。。。

```python
from pwn import *
from hashlib import sha256
from os import urandom
import socketserver
import signal
import random
import string

PREFIX = b"CryptoUserInfo"
random.seed(urandom(32))
def lazysend(s):
    io.sendline(s)
    time.sleep(0.05)


def get_token(byte: bytes) -> str:
    return sha256(PREFIX + byte).hexdigest()[-14:]

alphabet = string.ascii_letters + string.digits

def g(l):
    if l == 1:
        for i in alphabet:
            yield i
    else:
        for s in g(l - 1):
            for i in alphabet:
                yield s + i

def force():
    users = {}
    for username in g(10):
        c = get_token(username.encode("utf-8"))
        if c in users:
            return username, users[c]
        else:
            users[c] = username

# a, b = force()
a, b = ("aaaaaaRjiD", "aaaaaam08V")
print(1243234)
print(a)
print(b)

io = remote("chal.thuctf.redbud.info", 50963)
lazysend("1")
lazysend(a)
```

```
lazysend(get_token(a.encode("utf-8")))
lazysend("1")
lazysend(b)
lazysend(get_token(b.encode("utf-8")))
lazysend("2")
lazysend(a)
lazysend(get_token(a.encode("utf-8")))
io.interactive()
```

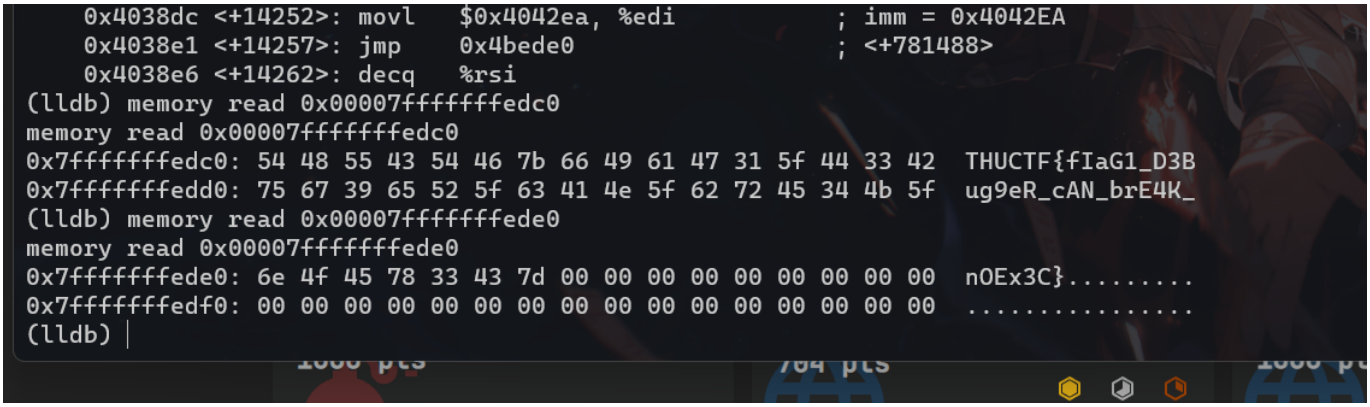
nc

`nc ip port`



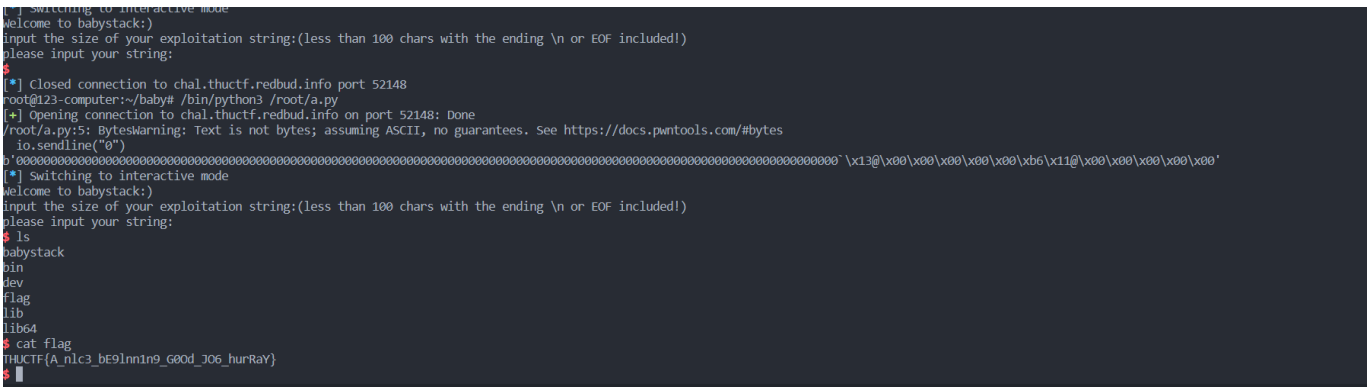
禁止执行，启动

观察后发现只有bin里的文件可以执行，然后看到了两个不是busybox创建的文件，一查发现是debugger，那随便debug一下busybox然后在rip处写入shellcode就行了。主要难点是现学lldb的语法~~以及修bug~~。（以及花了1个小时意识到548是十进制。。。）



babystack

IDA打开后可以发现一个后门函数，然后注意到长度判断是用无符号写的，所以可以输入0后栈溢出，依靠提示可以知道要把栈变为16的倍数才能调用后门，所以多写一个空的ret即可。



初学C语言

格式化字符串漏洞，所以只要疯狂%p%x然后看看里面哪些像ascii就行（问就是我也不会）

简单的打字稿

~~用于推销typescript的题。~~

typescript的类型不会在编译结果中出现，所以只能依靠报错来获取flag。推荐的方法是用函数传参来做，这样不需要去编一个实例出来。第一题简单传点数字就行。第二题需要将flag的值一层层提取出来，所以在网上查到了把union变intersection，获取返回值，获取参数类型的模板，然后一层层套用就行了。但是intersection变union的模板没找到，好在报错够短能过，就不管了。

```
function as(a:flag1){
}
as("2134124")
```

Su

```
Process exited with code 1
[+] Stdout:

[+] Stderr:
Check file:///app/$deno$stdin.ts
error: TS2345 [ERROR]: Argument of type '"2134124"' is not assignable to paramet
er of type '"THUCTF{t00O_3ASy_for_T00o_E4Sy_lANG}"'.
as("2134124")
    ~~~~~~~~~~
    at file:///app/$deno$stdin.ts:6:4
```

```
function s(xxx:UnionToIntersection<flag2>){
var t = new xxx().v
type c = ReturnType<typeof t>
type d  =ArgumentTypes<c>[0]
type e =ArgumentTypes<d>[1]
type f = keyof e
function rt(uu:e){
}
rt(123)
}
```

Submit!

```
Process exited with code 1
[+] Stdout:

[+] Stderr:
Check file:///app/$deno$stdin.ts
error: TS2345 [ERROR]: Argument of type 'number' is not assignable to parameter
of type '{ 'THUCTF{tYP35cRiPt_6EttEr_7han_pYthoN!}': never; } & Record<string, s
tring>'.
  Type 'number' is not assignable to type '{ 'THUCTF{tYP35cRiPt_6EttEr_7han_pYth
ON!}': never; }'.
rt(123)
   ~~~
```

(一些趣事)

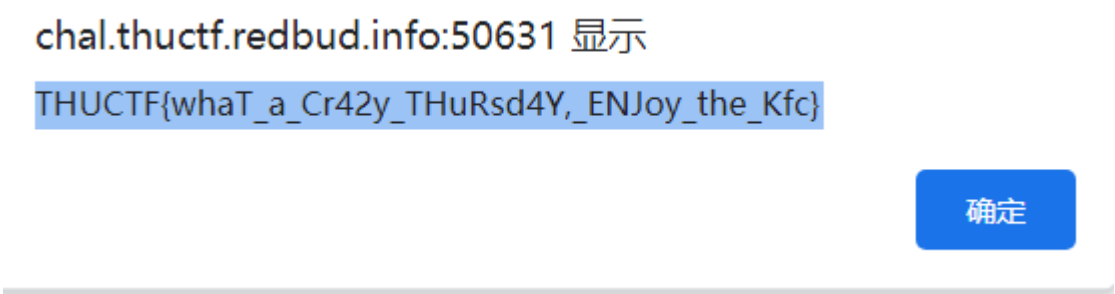
本来没打算做第二问，但是在翻选手博客时发现了一篇关于typescript类型的文章，于是去做了，虽然这篇文章对最终做出来的答案没啥帮助（）

Chrone1

本来对问号后面到底怎么写才能改hostname没有一点想法，但是提示实在是太强了，直接塞爆然后报错就可以了。

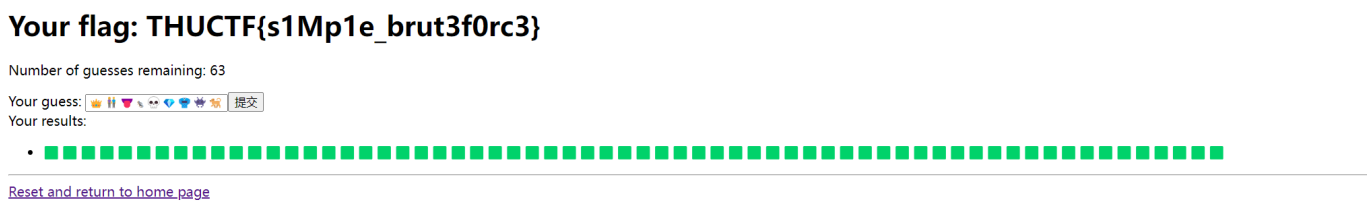
V ME 50

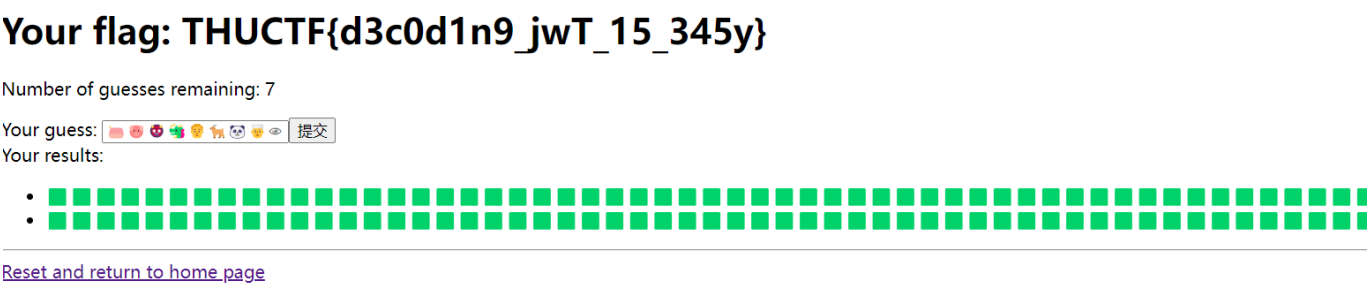
反复观察devtools里面的东西，可以看到一个被注释的权限修改，进去后提交不通过，发现有一个隐藏的id栏，修改value为1后过了。进入管理员发现可以买kfc和flag，然后换了一个账号发现订单没有变，于是只需要开10个账号就可以退款买flag了。



emodle

做这道题很明显靠js是不太够的，所以用了selenium来暴力搜索。第二问限制到了8次就没法解了，所以尝试去攻击session，结果session是用base64编码的，而且答案还在里面，直接抄就行了。（后来才知道这东西叫JWT）第三问删去了答案，且需要在1分钟内做完。但是注意到每次提交session都会更新，于是把session改回去后发现还可以玩，于是只要selenium继续暴力就行了。（selenium真好用）





Microsoft Edge 正由自动测试软件控制。

polynomials

纯逆向题。第一题可以很快发现是一个矩阵在F_p下求逆，然后上网抄代码（）。第二题是一堆奇怪玩意看不懂，选择手动写逆运算，然后看着那些常数觉得很怪，结果解出来发现是NTT。。。第三问吸取了教训，知道是多项式乘法，直接除回去。

```
import numpy
from sympy import Matrix, mod_inverse
import base64

p = 998244353


def matInvMod(vmnp, mod):
    nr = vmnp.shape[0]
    nc = vmnp.shape[1]
    if nr != nc:
        print("Error: Non square matrix! exiting")
        exit()
    vmsym = Matrix(vmnp)
    vmsymInv = vmsym.inv_mod(mod)
    vmnpInv = numpy.array(vmsymInv)
    k = nr
    vmtest = [[1 for i in range(k)] for j in range(k)]  # just a 2-d list
    vmtestInv = vmsym * vmsymInv
    for i in range(k):
        for j in range(k):
            # print i, j, vmtrx2[i,j] % mod
            vmtest[i][j] = vmtestInv[i, j] % mod
    print("test vmk*vkinv % mod \n:", vmtest)
    return vmnpInv


def reverse1():
    # p = 271
    k = 38
    a = [
        9403659313894707252,
        10993934582569577013,
        7795846728707237000,
        9191877155529135904,
        9908595251531158310,
        8660794244482574322,
        9874644655622856331,
        11162974483227675571,
        8812079262269580364,
        7753715510343656484,
        8903987252496368507,
        7868707299292224701,
        10611895060740010939,
        8123730605949078930,
        9609585807820621655,
        10265959052407839358,
```

```
        9267314521215120029,
        8352173460569731505,
        9303887510029974835,
        8883709811136466809,
        7879371872137715806,
        6849904959337600240,
        7512572632694415070,
        9819292060154201339,
        9002269748658262723,
        7492430953564765751,
        10110377915354936043,
        10141655795860606204,
        9164981623634819530,
        8972211292508642404,
        9705815770843805343,
        9917308592076637025,
        9178840956744520441,
        8757239147483357276,
        8363174767189842779,
        6905656485193533262,
        8306059725396726094,
        9827326457780163779,
    ]

    for t in a:
        print(hex(t % p)[2:])

    vvv = numpy.array(
        [
            0x0CB0,
            0x168C83AC,
            0x0D1D79D4,
            0x228A0DD,
            0x0E57451,
            0x25F3BF43,
            0x0F1653F7,
            0x395B969F,
            0x37198928,
            0x1651D179,
            0x20F1DF11,
            0x38F4DC2B,
            0x37CDD474,
            0x2043323C,
            0x0E4CB532,
            0x14FE0ADA,
            0x2DADCE9D,
            0x2C325FFB,
            0x0D9357C,
```

```
            0x1C90D4E6,
            0x19A7E972,
            0x24EAABA9,
            0x2C2A70ED,
            0x315995C6,
            0x1E48BE27,
            0x99C05B0,
            0x0EE775B0,
            0x27F52AA6,
            0x136F26DB,
            0x5CE66CF,
            0x37F9958D,
            0x2D634F37,
            0x0F424CE3,
            0x2348C868,
            0x0A16629F,
            0x2ACC2B38,
            0x0F7FEB61,
            0x159215F5,
        ]
    )
    vm = numpy.array([[(j + 1) ** i for i in range(k)] for j in range(k)])
    # vminv = modMatInv(vm, p)
    print(vm.dot(vvv))
    vminv = matInvMod(vm, p)
    print(vminv)
    vmtestnp = vm.dot(vminv) % p  # test mtrx inversion
    print(vmtestnp)
    print((vminv.dot(vvv)) % p)


def reverse2():
    output =
"zA8AAAQBAACR20kH6CQ2NCnKEw1xIConw/anNmUeGgw50hofiOzzAYcLCgLRq8M2MZBVBfSzxDRVgX
ASOMUYDPKa/Cqp7+oRmFmLCI/9wCwkCjcQo9LGCQWL0CmU1/QETk5fLASDAzoeuXEglllEG/U8NzHpb
egh+CG/N/s0kS8FBXchrTF6Ape+QxD5v4QMkWgoLk4FqCcS3oY4hzPgIO8k/hv1nIMBEq9iJUSfAAk7
Skso7HCqLqS7WQhBzAcV5cI0O/MZWCoioRoqs6HIFefUlCscB2A3r8NjLr0QXjE8UFQLjkD0Bj4NQAk
Aj7g4AwttMwTMTRbxvdsuNeJTDg=="
    output = base64.b64decode(output)
    vvv = []

    info = [
        0x00000000,
        0x00000001,
        0x00000001,
        0x3656D65B,
        0x00000001,
        0x163456B8,
```

```
        0x3656D65B,
        0x1D21561B,
        0x00000001,
        0x375FE6C1,
        0x163456B8,
        0x257C787F,
        0x3656D65B,
        0x16400573,
        0x1D21561B,
        0x2766E2AB,
        0x00000001,
        0x1AFD27AC,
        0x375FE6C1,
        0x27B55371,
        0x163456B8,
        0x0A25E8C8,
        0x257C787F,
        0x337E65BE,
        0x3656D65B,
        0x24C90037,
        0x16400573,
        0x20677ED8,
        0x1D21561B,
        0x267C5B5F,
        0x2766E2AB,
        0x3647FC39,
        0x00000001,
        0x3700CCCC,
        0x1AFD27AC,
        0x00E5B307,
        0x375FE6C1,
        0x131D28F6,
        0x27B55371,
        0x13477C50,
        0x163456B8,
        0x0448FFEC,
        0x0A25E8C8,
        0x16D34EAF,
        0x257C787F,
        0x320E0843,
        0x337E65BE,
        0x1132615E,
        0x3656D65B,
        0x0647D7A5,
        0x24C90037,
        0x0AD6B6CD,
        0x16400573,
        0x2ACA743E,
```

```
        0x20677ED8,
        0x0498A9B2,
        0x1D21561B,
        0x1CC06735,
        0x267C5B5F,
        0x17D4C6BF,
        0x2766E2AB,
        0x06B059A5,
        0x3647FC39,
        0x172CA754,
        0x00000001,
        0x2E97FC55,
        0x3700CCCC,
        0x341E50E0,
        0x1AFD27AC,
        0x036D036A,
        0x00E5B307,
        0x126C756C,
        0x375FE6C1,
        0x13F7A236,
        0x131D28F6,
        0x0922FC32,
        0x27B55371,
        0x35669A1A,
        0x13477C50,
        0x32321C0E,
    ]

    for i in range(len(output) // 4):
        vvv.append(int.from_bytes(output[4 * i : 4 * i + 4],
byteorder="little"))
    print(vvv)

    v9 = 1
    div2 = mod_inverse(2, p)
    while v9 < 64:
        i = ((63) // (2 * v9)) * 2 * v9
        while i >= 0:
            j = v9 - 1
            while j >= 0:
                v8 = (vvv[i + j + v9] * mod_inverse(info[j + v9], p)) % p
                vc = vvv[i + j]
                vvv[i + j] = ((v8 + vc) * div2) % p
                vvv[i + j + v9] = ((v8 - vc) * div2) % p
                j -= 1
            i -= 2 * v9
        v9 *= 2
    print(vvv)
```

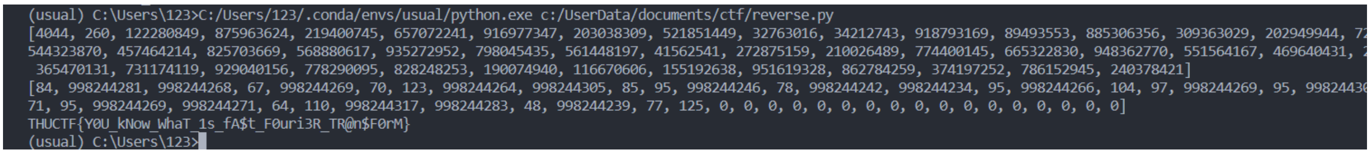
```
    for i in vvv:
        if i > 128:
            i = p - i
        print(chr(i), end="")


def reverse3():
    output =
"DCcAAJxCAABbZwAAin8AAJulAADNwQAAy/YAAAIOAQCILwEAi0wBACVVAQAkiQEAmKMBAEW1AQCvwg
EAjAECANT9AQAzMwIAaGACAFeVAgDvigIAcqgCAOTmAgBe7gIAnQoDAO4MAwDAPgMAmmIDAId0AwBOn
QMA9+IDAM/sAwBLMAQAZF0EAF5xBAB0xAQAXuoEAIHwBAAp/gQAViYFAME9BQBaNQUAtUsFAAxWBQBs
KQUAPlIFALpMBQAwOQUAXyYFAPJSBQBqMQUAlT4FACNYBQC3ZAUA3CwFACgSBQAoXwUAWE0FAEtFBQB
XLwUAb00FABE+BQCRXwUAUFoFAA=="
    output = base64.b64decode(output)
    input = "welcome to the world of polynomial"
    ptr = [0] * 64
    for i in range(64):
        ptr[i] = ord(input[i % 34])
    vvv = [0] * 128
    divin = mod_inverse(990445569, p)

    for i in range(len(output) // 4):
        vvv[i] = int.from_bytes(output[4 * i : 4 * i + 4], byteorder="little")

    print(vvv)
    print(ptr)
    sss = []
    divp = mod_inverse(ptr[0], p)
    for i in range(64):
        c = (vvv[i] * divp) % p
        for j in range(64):
            vvv[i + j] = (vvv[i + j] - c * ptr[j]) % p
        print(c)
        print(vvv)
        sss.append(chr(c))
    print("".join(sss))
```

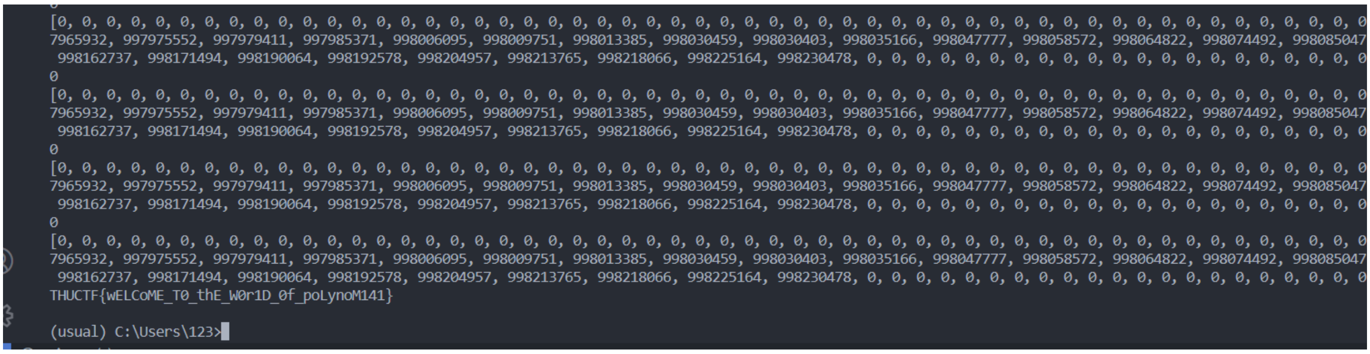
```
50          7879371872137715806,
51          6849904959337600240,
52          7512572632694415070,
53          9819292060154201339,
54          9002269748658262723,
55          7492430953564765751,
56          10110377915354936043,
57          10141655795860606204,
58          9164981623634819530,
59          8972211292508642404,
```

```
        9705815770843805343,
        9917308592076637025,
        9178840956744520441,
        8757239147483357276,
        8363174767189842779,
        6905656485193533262,
        8306059725396726094,
        9827326457780163779,
    ]

    for t in a:
        print(hex(t % p)[2:])

    vvv = numpy.array(
        [
            0x0CB0,
            0x168C83AC,
            0x0D1D79D4,
```

问题 输出 调试控制台 终端 端口

```
5f
6d
41
73
37
45
72
5f
4f
46
5f
6c
41
36
52
34
6e
67
65
7d
[18069583275 104287957166057878256 245181954600606599129141567
 901472765531646749542253949667 2 323830247612226980124189728305521 23
 263929888634218373346610206975688930 40
```





扫雷

重量级题目。

简单玩玩就可以发现它有一个几乎固定的边框，而且里面的大块十分的方正，除了少数同一种特殊玩意。然后写脚本把所有大块都先点了。然后发现解不动了，于是去看右边的3个一组的玩意，折磨了好久，最后解出来是一个3-SAT。。。知道了就好办了，直接枚举就可以确定唯一解，然后让脚本把关键块点出来后疯狂右键就行了。

脚本采用了注入了render的方法来进行回调，然后就可以写了，甚至没有js压缩（）

```
async function sleep(t) {
    return new Promise((resolve) => { setTimeout(resolve, t) })
}


function renderboard() {
    if (!checkready()) {
        setTimeout(renderboard, 100);
        return;
    }
    console.log("rendering");
    let boardx = boardobj.sizex;
    let boardy = boardobj.sizey;

    if (boardobj.type == 0) {
        board = [];
        for (let i = 0; i < boardx; i++) {
            board[i] = boardobj.board[i].split("");
        }
    }
    else {
        for (let i = 0; i < boardobj.board.length; i++) {
            let x = boardobj.board[i][0];
            let y = boardobj.board[i][1];
            let st = boardobj.board[i][2];
            board[x][y] = st;
        }
    }

    if (draw) {
        for (let i = 0; i < boardx; i++) {
            for (let j = 0; j < boardy; j++) {
                let st = board[i][j];
                if (lboard != undefined && lboard[i][j] == st) {
                    continue;
                }
                let id = 0;
                if (st == 'F')
                    id = 10;
                else if (st == '*')
                    id = 11;
                else if (st == '.')
                    id = 9;
                else
                    id = parseInt(st);
                ctx.drawImage(images[id], j * TILE_SIZE, i * TILE_SIZE,
TILE_SIZE, TILE_SIZE);
```

```
            }
        }
        lboard = JSON.parse(JSON.stringify(board));
    }
    find()
}

function find() {
    if (first) {
        first = false
        callback()

    }
}

async function detect_times(t) {
    for (var i = 0; i < t; i++) {
        await run(0, 0, 'detect')
        console.log(i + 1)
    }
}

var callback = () => { }
var first = true
var draw = 1

async function run(x, y, type = 'open') {
    if (x >= boardobj.sizex || y >= boardobj.sizey) {
        return new Promise((resolve) => { resolve() })
    } if (type == 'open' && board[x][y] != '.') {
        return new Promise((resolve) => { resolve() })
    }
    first = true
    var c = new Promise((resolve) => { callback = resolve })
    if (type == 'open') {
        doopen(x, y)
        console.log('run doopen(' + x + ', ' + y + ')')
    }
    if (type == 'detect') {
        dodetect()
        console.log('run dodetect()')
    }
    return c
}

async function main() {
    await run(0, 0)
    let boardx = boardobj.sizex;
```

```
    let boardy = boardobj.sizey;
    locationy = new Array()
    for (let i = 1; i < boardy; i++) {
        if (board[1][i] == '3') {
            locationy.push(i)
        }
    }
    console.log(locationy)
    draw = 0
    bin = []
    for (let i = 0; i < locationy.length; i++) {
        column = locationy[i] + 2
        for (var j = 4; j < boardx; j++) {
            var c = board[j][column]
            if (c == '0')
                continue
            if (c == '.')
                continue
            if (j + 1 < boardx && board[j + 1][column] == '.') {
                if (c == '1') {
                    await run(j + 3, column)
                }
                if (c == '2') {
                    await run(j + 1, column - 1)
                    await run(j + 1, column)
                    await run(j + 2, column)
                    await run(j + 3, column)
                    await run(j + 3, column + 2)
                    await run(j + 4, column)
                    await run(Math.floor(j / 3) * 3 - 1, column - 2)
                    bin[j + 2] = column
                    await run(j + 6, column)
                }
            }
        }
    }
    draw = 1
    console.log(bin)
    await run(0, 0, 'detect')
    await update()
}

async function update() {
    await run(0, 0)
    let boardx = boardobj.sizex;
    let boardy = boardobj.sizey;
    locationy = new Array()
    for (let i = 1; i < boardy; i++) {
```

```
        if (board[1][i] == '3') {
            locationy.push(i)
        }
    }
    locationx = new Array()
    for (let j = 1; j < boardx - 1; j++) {
        if (board[j + 1][113] == "1" && board[j - 1][113] == "1") {
            locationx.push(j)
        }
    }
    // console.log(locationx)
    // console.log(locationy)
    mark = new Array()
    for (var i = 0; i < locationx.length; i++) {
        for (var j = 0; j < locationy.length; j++) {
            if (board[locationx[i] + 1][locationy[j] - 3] == "3") {
                if (board[locationx[i] + 2][locationy[j] - 3] == "2" &&
board[locationx[i]][locationy[j] - 3] == "2") {
                    //console.log(i, j)
                    mark[i] = j
                }
            }
        }
    }
    neg = new Array()
    for (var i = 0; i < locationx.length; i++) {
        neg[i] = board[locationx[i] + 1][117] == "3"
    }
    console.log(mark)
    console.log(neg)
    var result = new Array();
    for (i = 0; i < (2 << locationy.length); i += 2) {
        var b = new Array()
        for (var j = 0; j < locationy.length; j++) {
            b[j] = Boolean(i & 2 << j)
        }
        var flag = 1;
        for (j = 0; j < locationx.length; j += 3) {
            c = (neg[j] ^ b[mark[j]])
            c = c || (neg[j + 1] ^ b[mark[j + 1]])
            c = c || (neg[j + 2] ^ b[mark[j + 2]])
            if (!c) {
                flag = 0;
                break
            }
        }
        if (flag) {
            result.push(b)
```

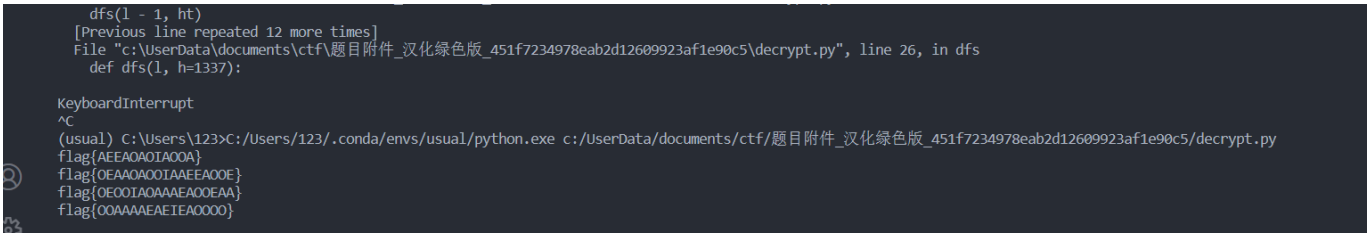
```
            console.log(i)
            // console.log(b)
        }
    }
    console.log(result)
    async function apply(r) {
        for (var i = 0; i < locationy.length; i++) {
            if (r[i]) {
                await run(4, locationy[i])
            }
            else {
                await run(3, locationy[i])
            }
        }
        for (var i = 0; i < locationx.length; i++) {
            await run(locationx[i], 112)
            if (r[mark[i]]) {
                await run(locationx[i], 113)
            }
            else {
                await run(locationx[i], 111)
            }
            if (i % 3 == 0) {
                await run(locationx[i], 138)
                await run(locationx[i], 126)
            }
        }
    }
    apply(result[0])
}

main()
```

```
53
run dodetect()
rendering
54
run dodetect()
rendering
55
run dodete
rendering
56
run dodete
rendering
57
run dodetect()
rendering
58
run dodetect()
THUCTF{jUs7_a_siMp13_m1neSwEeP3R_saT_pro61em}
rendering
59
run dodetect()
THUCTF{jUs7_a_siMp13_m1neSwEeP3R_saT_pro61em}
rendering
60
run dodetect()
>
```

汉化！

~~还好我之前拆过~~。~~去~~网上找kirikiri2的解包软件即可。打开在最后一个场景就可以发现flag（我被骗了）第二问需要研究它是怎么判断相等的，发现是一个哈希，但爆破这个哈希会有很多解，于是继续去拆存档，注意到除了data0以外的另外两个文件也是可以拆的，然后就发现了游戏所有选项的选择次数，于是就加上限制条件继续枚举就解出来了。

```
xp= f.hash == f.prev_hash

torage="66ccff"
确！大概。[l][r]
合，Flag 1 是：[font color="0x66ccff" size=0]flag{did-you-unpack-the-xp3?}
tfont]。[l][r]
2 是出题人在存档里输入的内容。[l][r]



 time=1000]
```



流量包

打开第二题的流量包，发现很多的ascii字符，于是去搜索了一下发现是一个古老的协议。用lrzsz链接即可。第二题需要去翻协议细节，然后在知乎上找到了一个实现的源码，然后用python抄一份就行（问就是不会写cmake）但是需要注意把源码的crc16换成crc32，不然就得自己在本地抓包修bug了。

```python
import os

raw = b""
# with open("flag2.jpg", "rb") as fp:
#     tran = fp.read()
with open("flag.jpg") as fp:
    tran = fp.read().replace("\n", "").replace(" ", "")
    tran = bytes.fromhex(tran)

eat_crc = 0
print(len(tran))
count = 0
for i in tran:
    if i == 24:
        count += 1
print(count)
traned = 0
tr = 0

for i in range(len(tran)):
    u = tran[i].to_bytes()
    if not traned:
        if eat_crc:
            eat_crc -= 1
            if tr:
                tr = 0
                continue
            if u == b"\x18":
                eat_crc += 1
                tr = 1
            continue
        if u != b"\x18":
            raw += u
            continue
        else:
            traned = 1
            continue
    traned = 0
    if u == b"\x69":
        eat_crc = 4
        continue
    if u == b"\x68":
        eat_crc = 2
        continue
    if u == b"\x6c":
        raw += b"\x7f"
        continue
```

```
    if u == b"\x6d":
        raw += b"\xff"
        continue
    if tran[i] & 0x60 == 0x40:
        raw += (tran[i] ^ 0x40).to_bytes()
    else:
        print(i)

# print(raw)
with open("flag3.jpg", "wb") as fp:
    fp.write(raw)
```

```
short options use the same arguments as the long ones
root@123-computer:~# rz --tcp-client chal.thuctf.redbud.info:51183 -vvv  --overwrite
rz 0.12.21rc

connecting to [chal.thuctf.redbud.info] <51183>

mode:1
rz waiting to receive.zshhdr: ZRINIT 23000000zgethdr: ZRQINIT 0zshhdr: ZRINIT 23000000zgethdr: ZFILE 0zrc
ode:3
zmanag=0, Lzmanag=0
zconv=1
Receiving: flag.txt
zshhdr: ZRPOS 0zgethdr: ZDATA 0zrdat32: 55 ZCRCEzgethdr: ZEOF 37rzfile: normal EOF
Bytes received:      55/     55   BPS:31056
zshhdr: ZRINIT 23000000zgethdr: ZFIN 0ackbibi:
zshhdr: ZFIN 0ackbibi complete
mode:0

Transfer complete
root@123-computer:~# ls
123             deploy                                      service_checker_695b0658917deedda173b43630
baby            easymaze_9d2ac5e585b784db5d8c7bca07728f36.png  snap
binwalk-master  flag.txt                                    vme50.py
darkroom.py     notebook_58fd1bf623f814e2767049f98a1345c4.py  wsl
root@123-computer:~# cat flag.txt
THUCTF{Anc1ent_tr4nsf3r_pr0tOcoI_15_57111_In_u5e_t0d4y}root@123-computer:~#
```

flag{traFf1c_aNa1y51s_4_ZMODEM}

喵

做这道题需要写一个简易编译器喵。

可以修改一下filtered实现debug喵。

第一题重复把字符换成emoji谢谢喵。

重复把10个emoji换成另一个emoji谢谢喵。

剩下的emoji就是个位数喵。

重复这件事喵。

第二题在每行后面加上长度喵。

对长度第一位做冒泡排序喵。

写一个简易栈来对第二位继续排序喵。

对需要排的和已经排好的位都标记一下喵。

然后就排好了喵

谢谢喵

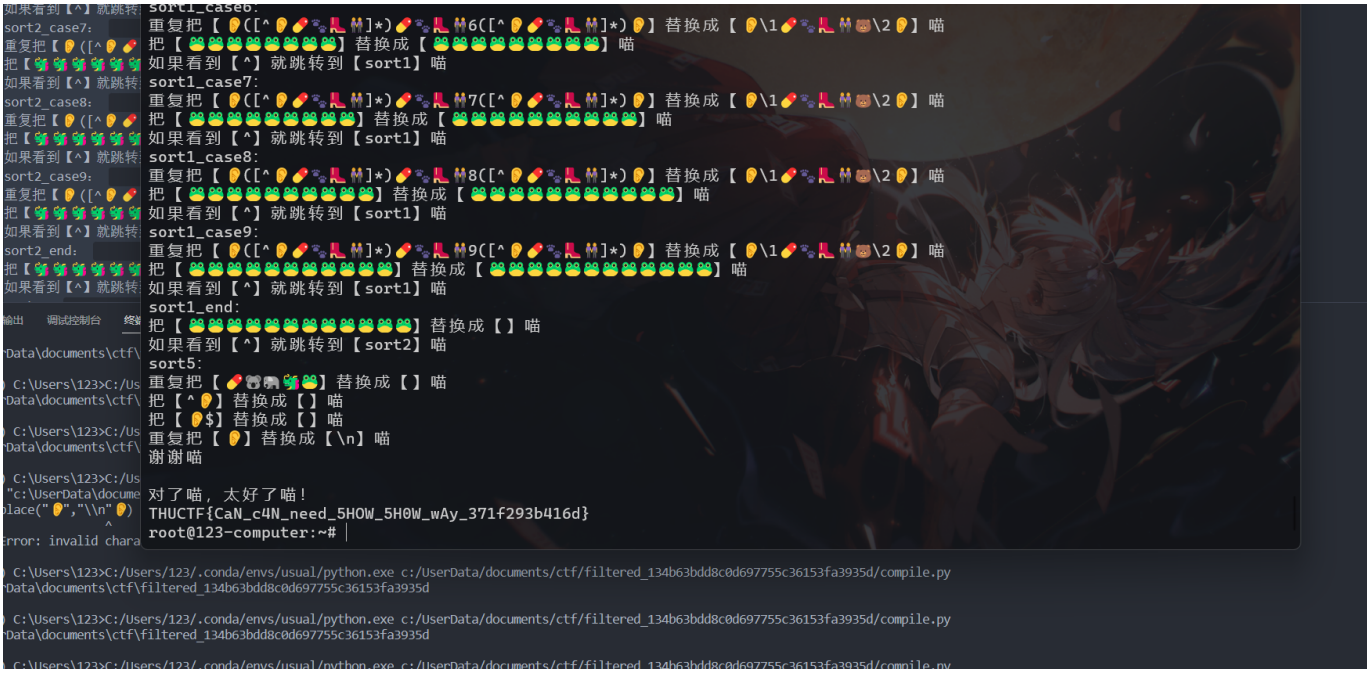
```
import re
from re import Pattern
from dataclasses import dataclass
from typing import TextIO
import os

inst = []

os.chdir(os.path.dirname(__file__))
print(os.getcwd())


class Inst:
    pass


@dataclass(frozen=True)
class Replace(Inst):
    repeat: bool
    regex: str
    to: str


@dataclass(frozen=True)
class Branch(Inst):
    neg: bool
    regex: str
    label: str


@dataclass(frozen=True)
class Label(Inst):
    value: str


def replace(reg, to, repeat=True):
    inst.append(Replace(bool(repeat), reg, to))


def branch(neg, reg, label):
    if reg == "":
        reg = "^"
    inst.append(Branch(bool(neg), reg, label))


def label(name):
    inst.append(Label(name))
```

```python
def compile():
    output = []
    for i in inst:
        s = ""
        if isinstance(i, Replace):
            if i.repeat:
                s = "重复"
            s += f"把【{i.regex}】替换成【{i.to}】喵"
        if isinstance(i, Branch):
            s = "如果"
            if i.neg:
                s += "没"
            s += f"看到【{i.regex}】就跳转到【{i.label}】喵"
        if isinstance(i, Label):
            s = i.value + ": "
        output.append(s)
        # print(s)
    output.append("谢谢喵")
    # print("谢谢喵")
    return output


def moew1():
    replace("[^👆]", "👆")
    replace("$", "👀", False)
    label("load")
    replace("👆" * 10, "👌")
    replace("👀", "👆👀", False)
    for i in range(10):
        replace("👆" * (10 - i) + "👀", "👀" + str(9 - i), False)
    replace("👌", "👆")
    branch(False, "👆", "load")
    replace("👀", "", False)


def moew2():
    replace("^", "👂", False)
    replace("$", "👂", False)
    replace("\\n", "👂")
    replace("👂👂", "👂")
    replace("👂([^👂💊]+)👂", "👂\\1💊\\1👂")
    replace("💊[^💊👆👂]([^👂💊]*)👂", "💊\\1👆👂")
    replace("💊(👆*)👂", "💊\\1👀👂")
    for i in range(4):
        replace("👆" * 10, "👌")
        replace("👀", "👆👀", False)
```

```
        for i in range(10):
            replace("👆" * (10 - i) + "👀", "👀" + str(9 - i), False)
        replace("👌", "👆")
    replace("👀", "", False)

    def sorti(t, recv, send, tmp):
        label(f"sort{t}")
        replace(recv + send, recv + tmp)
        branch(False, tmp * 12, f"sort{t}_end")
        for i in range(10):
            branch(False, tmp * (11 - i), f"sort{t}_case{9-i}")
        label(f"sort{t}_default")
        for i in range(10):
            replace(f"👂([^👀{tmp}]*)👂([^👂👀{recv}{tmp}]*{recv}{i}[^👀👂
{recv}]*)👂", "👂\\2👀👂\\1👂")
            replace(f"👂([^👂👀{recv}]*{recv}{i}[^👀👂{recv}]*)👂", "👂\\1👀👂",
False)
        replace("👀", "", False)
        replace("$", tmp * 2, False)
        branch(False, "", f"sort{t}")
        for i in range(10):
            label(f"sort{t}_case{i}")
            replace(f"👂([^👂{recv}]*){recv}{i}([^👂{recv}]*)👂", f"👂\\1{recv}
{send}\\2👂")
            replace(tmp * (i + 2), tmp * (i + 3), False)
            if t > 1:
                branch(False, "", f"sort{t-1}")
            else:
                branch(False, "", f"sort{t}")
        label(f"sort{t}_end")
        replace(tmp*12,"",False)
        branch(False, "", f"sort{t+1}")

    global s
    global v
    s = "💊"
    v = s
    def sortii(t,a,b):
        global s
        global v
        v+=b
        sorti(t,s,a,b)
        s+=a
    # sortii(5,"👸","🐋",)
    sortii(4,"🐾","🐶")
    sortii(3,"👢","🦏")
    sortii(2, "👫","🐲" )
    sortii(1, "🐻", "🐸")
```

```
    label("sort5")
    print(v)
    replace(v,"")
    replace("^👂","",False)
    replace("👂$","",False)
    replace("👂","\\n")

    # for i in range(10):
    #     replace(f"👂([^👂👀💊]+)💊{i}([^👀 💊 ) 👂", "👂\\1👆💊\\2👀👂", False)
    #     Branch(False, "", "sort4")
    # label("sort4")
    # for i in range(10):
    #     replace(f"👂([^👀]*)👂([^👂👀💊]+👆💊{[}^ 👂   💊] + )👂", "👂\\2👀👂\\1
👂")
    #     replace(f"👂([^👂👀💊]+👆💊{i}[^👀👂💊] +) 👂", "👂\\1👀👂", False)
    # replace("👀", "", False)


moew2()

s=""
v=""
out = compile()
with open("input.txt", "w", encoding="utf-8") as fp:
    for s in out:
        fp.write(s)
        fp.write("\n")
```

```
把【$】替换成【👀】喵
load:
重复把【👆👆👆👆👆👆👆👆👆👆】替换成【✌】喵
把【👀】替换成【👆👀】喵
把【👆👆👆👆👆👆👆👆👆👀】替换成【👀9】喵
把【👆👆👆👆👆👆👆👆👀】替换成【👀8】喵
把【👆👆👆👆👆👆👆👀】替换成【👀7】喵
把【👆👆👆👆👆👆👀】替换成【👀6】喵
把【👆👆👆👆👆👀】替换成【👀5】喵
把【👆👆👆👆👀】替换成【👀4】喵
把【👆👆👆👀】替换成【👀3】喵
把【👆👆👀】替换成【👀2】喵
把【👆👀】替换成【👀1】喵
把【👆👀】替换成【👀0】喵
重复把【✌】替换成【👆】喵
如果看到【👆】就跳转到【load】喵
把【👀】替换成【】喵
谢谢喵
对了喵，太好了喵！
THUCTF{WOw_Y0u_c4n_r3AIIY_R3g3X_9fa05114390b}
```


sort1_case6:
重复把【👂([^👂🧨🐾👢👫]*)🧨🐾👢👫6([^👂🧨🐾👢👫]*)👂】替换成【👂\1🧨🐾👢👫🐻\2👂】喵
把【🐸🐸🐸🐸🐸🐸🐸🐸🐸】替换成【🐸🐸🐸🐸🐸🐸🐸🐸🐸】喵
如果看到【^】就跳转到【sort1】喵
sort1_case7:
重复把【👂([^👂🧨🐾👢👫]*)🧨🐾👢👫7([^👂🧨🐾👢👫]*)👂】替换成【👂\1🧨🐾👢👫🐻\2👂】喵
把【🐸🐸🐸🐸🐸🐸🐸🐸】替换成【🐸🐸🐸🐸🐸🐸🐸🐸】喵
如果看到【^】就跳转到【sort1】喵
sort1_case8:
重复把【👂([^👂🧨🐾👢👫]*)🧨🐾👢👫8([^👂🧨🐾👢👫]*)👂】替换成【👂\1🧨🐾👢👫🐻\2👂】喵
把【🐸🐸🐸🐸🐸🐸🐸🐸🐸】替换成【🐸🐸🐸🐸🐸🐸🐸🐸🐸】喵
如果看到【^】就跳转到【sort1】喵
sort1_case9:
重复把【👂([^👂🧨🐾👢👫]*)🧨🐾👢👫9([^👂🧨🐾👢👫]*)👂】替换成【👂\1🧨🐾👢👫🐻\2👂】喵
把【🐸🐸🐸🐸🐸🐸🐸🐸🐸🐸】替换成【🐸🐸🐸🐸🐸🐸🐸🐸🐸🐸】喵
如果看到【^】就跳转到【sort1】喵
sort1_end:
把【🐸🐸🐸🐸🐸🐸🐸🐸🐸🐸🐸】替换成【】喵
如果看到【^】就跳转到【sort2】喵
sort5:
重复把【🧨🐻🎮🐸🐸】替换成【】喵
把【^👂】替换成【】喵
把【👂$】替换成【】喵
重复把【👂】替换成【\n】喵
谢谢喵
对了喵，太好了喵！
THUCTF{CaN_c4N_need_5HOW_5H0W_wAy_371f293b416d}
root@123-computer:~#