网刃杯复现misc+流量包

原创

@Demo 于 2021-09-14 09:55:23 发布 222 收藏

分类专栏： misc 文章标签： c# 自动驾驶 自然语言处理

本文链接：https://blog.csdn.net/qq_47886905/article/details/120272377

misc 专栏收录该内容

9 篇文章 0 订阅

订阅专栏

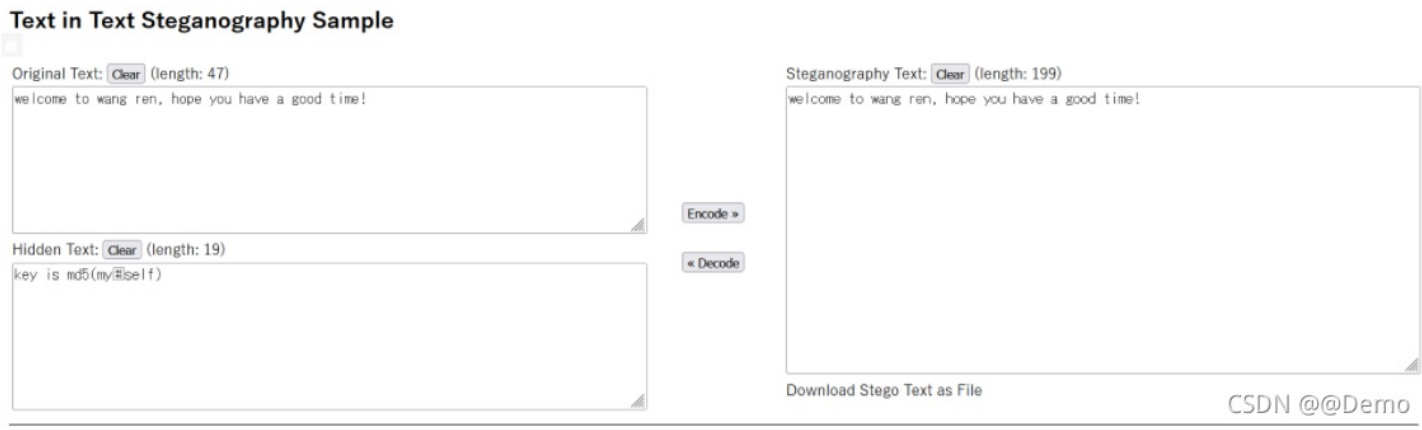
签到

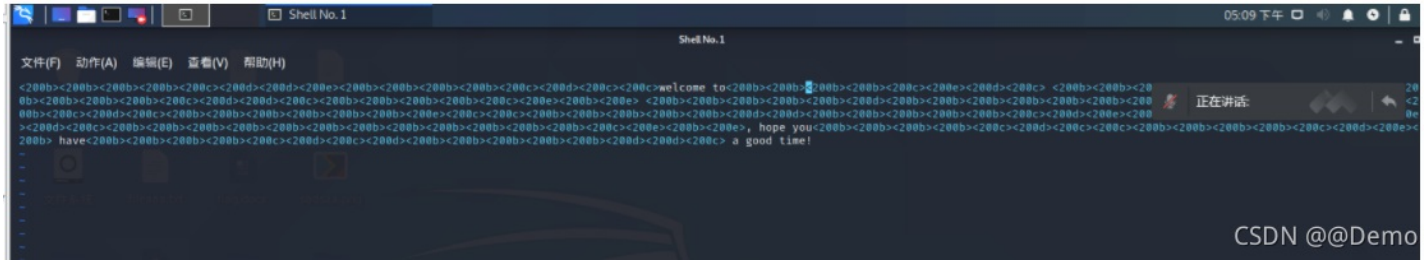
cipher.txt

```
U2FsdGVkX1+WTSHujcCjvHj/gcwL0C7u37XtW4idGcpci3H913I=
```

U2F那些，考虑为aes,des啥的

发现flag.txt是零宽度字符，





md5加密一下文件

```
f71b6b842d2f0760c3ef74911ffc7fdb
```

测试发现是rabbit，解密获得flag

mspaint

```
root@kali:~/volatility_2.6_lin64_standalone# volatility -f mspaint.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Wi
Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/root/volatility_2.6_lin64_standalone/mspaint.ra
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf8000403c0a0L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff8000403dd00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2021-09-08 09:47:28 UTC+0000
     Image local date and time : 2021-09-08 17:47:28 +0800
```


看一下iehistory

```
**************************************************
Process: 2440 explorer.exe
Cache type "URL " at 0x3606d00
Record length: 0x100
Location: Visited: qiyue@https://pan.baidu.com/s/1ZobcXkTaKm09R_nTlOno2w
Last modified: 2021-09-08 09:45:55 UTC+0000
Last accessed: 2021-09-08 09:45:55 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0xa8
**************************************************
Process: 2440 explorer.exe
Cache type "URL " at 0x3606e00
Record length: 0x180
Location: Visited: qiyue@https://pan.baidu.com/share/init?surl=ZobcXkTaKm09R_nTlOno2w
Last modified: 2021-09-08 09:46:13 UTC+0000
Last accessed: 2021-09-08 09:46:13 UTC+0000
File Offset: 0x180, Data Offset: 0x0, Data Length: 0xb4
**************************************************
Process: 2440 explorer.exe
Cache type "URL " at 0x3606f80
Record length: 0x100
Location: Visited: qiyue@file:///C:/Program%20Files/Common%20Files/key.png
Last modified: 2021-09-08 09:47:01 UTC+0000
Last accessed: 2021-09-08 09:47:01 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0xac
**************************************************
```

发现key.png和白度网盘

```
**************************************************
Process: 3568 iexplore.exe
Cache type "URL " at 0x3db5000
Record length: 0x100
Location: :2021090820210909: qiyue@https://pan.baidu.com/share/init?surl=ZobcXkTaKm09R_nTlOno2w
Last modified: 2021-09-08 17:46:13 UTC+0000
Last accessed: 2021-09-08 09:46:13 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x0
**************************************************
Process: 3568 iexplore.exe
Cache type "URL " at 0x3db5100
Record length: 0x100
Location: :2021090820210909: qiyue@:Host: pan.baidu.com
Last modified: 2021-09-08 17:45:58 UTC+0000
Last accessed: 2021-09-08 09:45:58 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x0
**************************************************
Process: 3568 iexplore.exe
Cache type "URL " at 0x3db5200
Record length: 0x100
Location: :2021090820210909: qiyue@https://pan.baidu.com/share/init?surl=ZobcXkTaKm09R_nTlOno2w
Last modified: 2021-09-08 17:46:12 UTC+0000
Last accessed: 2021-09-08 09:46:12 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x0
```

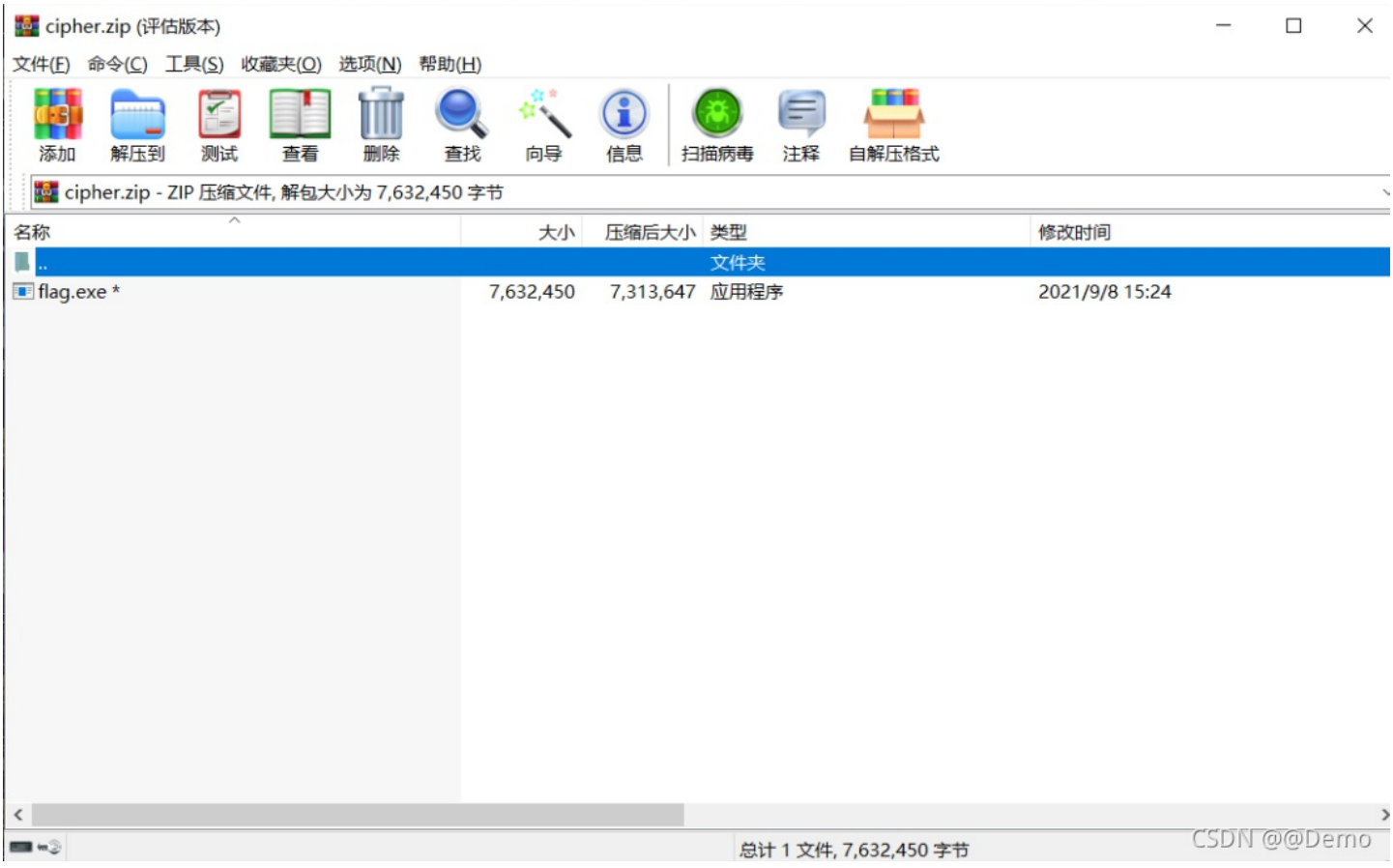


```
volatility -f mspaint.raw --profile=Win7SP1x64 filescan | grep key.png
```

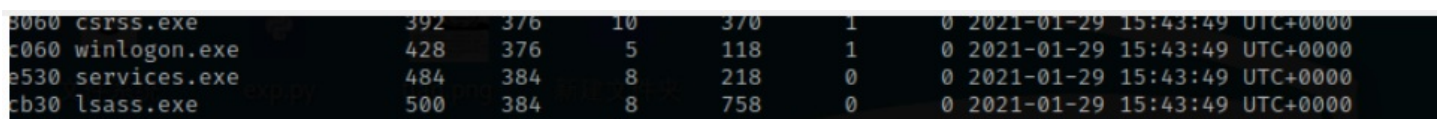
```
Volatility Foundation Volatility Framework 2.6
0x000000003e96e7d0      2      0 -W-rwd \Device\HarddiskVolume2\Program Files\Common Files\key.pngare-qiyue\VMwareDnD\20566876\key.png
```

```
volatility -f mspaint.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000003e96e7d0  -D ./
```

得到百度网盘的密码



发现加密的压缩包，还需要key



```
0b30 lsm.exe                508    384     11      145      0      0 2021-01-29 15:43:49 UTC+0000
d670 svchost.exe            616    484     12      371      0      0 2021-01-29 15:43:50 UTC+0000
4060 vmacthlp.exe           676    484      3       53      0      0 2021-01-29 15:43:50 UTC+0000
5b30 svchost.exe            720    484     10      322      0      0 2021-01-29 15:43:50 UTC+0000
c890 svchost.exe            796    484     21      593      0      0 2021-01-29 15:43:50 UTC+0000
5890 svchost.exe            856    484     23      464      0      0 2021-01-29 15:43:50 UTC+0000
f750 svchost.exe            884    484     31     1016      0      0 2021-01-29 15:43:50 UTC+0000
f5f0 svchost.exe            272    484     15      428      0      0 2021-01-29 15:43:51 UTC+0000
fb30 svchost.exe            788    484     18      506      0      0 2021-01-29 15:43:51 UTC+0000
c220 spoolsv.exe           1156    484     13      278      0      0 2021-01-29 15:43:51 UTC+0000
a830 svchost.exe           1184    484     19      325      0      0 2021-01-29 15:43:51 UTC+0000
a060 svchost.exe           1280    484     20      307      0      0 2021-01-29 15:43:52 UTC+0000
db30 VGAuthService.        1340    484      3       86      0      0 2021-01-29 15:43:52 UTC+0000
c060 vmtoolsd.exe          1456    484      9      267      0      0 2021-01-29 15:43:52 UTC+0000
bb30 svchost.exe           1632    484      6       93      0      0 2021-01-29 15:43:52 UTC+0000
d960 msdtc.exe             1944    484     12      144      0      0 2021-01-29 15:43:53 UTC+0000
cb30 WmiPrvSE.exe          1332    616     10      212      0      0 2021-01-29 15:43:54 UTC+0000
9830 taskhost.exe          2320    484      9      218      1      0 2021-01-29 15:45:19 UTC+0000
9730 dwm.exe               2400    856      5      236      1      0 2021-01-29 15:45:19 UTC+0000
5b30 explorer.exe          2440   2392     46     1207      1      0 2021-01-29 15:45:19 UTC+0000
e530 vmtoolsd.exe          2584   2440      9      223      1      0 2021-01-29 15:45:20 UTC+0000
f060 SearchIndexer.        2788    484     14      791      0      0 2021-01-29 15:45:26 UTC+0000
fb30 wmpnetwk.exe          2916    484     15      465      0      0 2021-01-29 15:45:26 UTC+0000
f060 svchost.exe           1444    484      4      266      0      0 2021-01-29 15:45:27 UTC+0000
5b30 sppsvc.exe            1404    484      4      152      0      0 2021-01-29 15:45:52 UTC+0000
eb30 svchost.exe           2892    484     14      324      0      0 2021-01-29 15:45:53 UTC+0000
24f0 cmd.exe               3316   2440      1       19      1      0 2021-09-08 07:35:47 UTC+0000
7060 conhost.exe           2684    392      2       59      1      0 2021-09-08 07:35:47 UTC+0000
3b30 mspaint.exe           2296   2440      5      119      1      0 2021-09-08 07:37:10 UTC+0000
f060 svchost.exe           2196    484      6      103      0      0 2021-09-08 07:37:10 UTC+0000
9b30 dllhost.exe           1524    616     11      179      1      0 2021-09-08 09:36:37 UTC+0000
8920 audiodg.exe           1824    796      5      132      0      0 2021-09-08 09:45:11 UTC+0000
55b0 iexplore.exe          2616   2440     16      607      1      1 2021-09-08 09:45:41 UTC+0000
6b30 iexplore.exe          3916   2616     25      682      1      1 2021-09-08 09:45:41 UTC+0000
5930 iexplore.exe          3568   2616     22      623      1      1 2021-09-08 09:45:51 UTC+0000
d190 SearchProtocol        1560   2788      8      340      0      0 2021-09-08 09:45:52 UTC+0000
7060 SearchFilterHo        3288   2788      6      140      0      0 2021-09-08 09:45:52 UTC+0000
7340 DumpIt.exe            1064   2440      1       25      1      1 2021-09-08 09:47:25 UTC+0000
0060 conhost.exe           1856    392      2       58      1      0 2021-09-08 09:47:25 UTC+0000
3920 dllhost.exe           3272    616      6       91      1      0 2021-09-08 09:47:25 UTC+0000
```



```
latility_2.6_lin64_standalone# volatility -f mspaint.raw --profile=Win7SP1x64 cmdscan
ndation Volatility Framework 2.6
**************************************
: conhost.exe Pid: 2684
: 0x1bfde0 Application: cmd.exe Flags: Allocated, Reset
3 LastAdded: 2 LastDisplayed: 2
0 CommandCountMax: 50
 0x5c
430: I am a painter
f70: My favorite thing is painting!!!
c00: I usually have a good habit of saving screenshots
61c0:
0158:
**************************************
: conhost.exe Pid: 1856
: 0x2afde0 Application: DumpIt.exe Flags: Allocated
0 LastAdded: -1 LastDisplayed: -1
0 CommandCountMax: 50
 0x5c
```


想到截屏

```
volatility -f mspaint.raw --profile=Win7SP1x64 screenshot -D ./
```

得到一个key，但是并不是压缩包的密码。

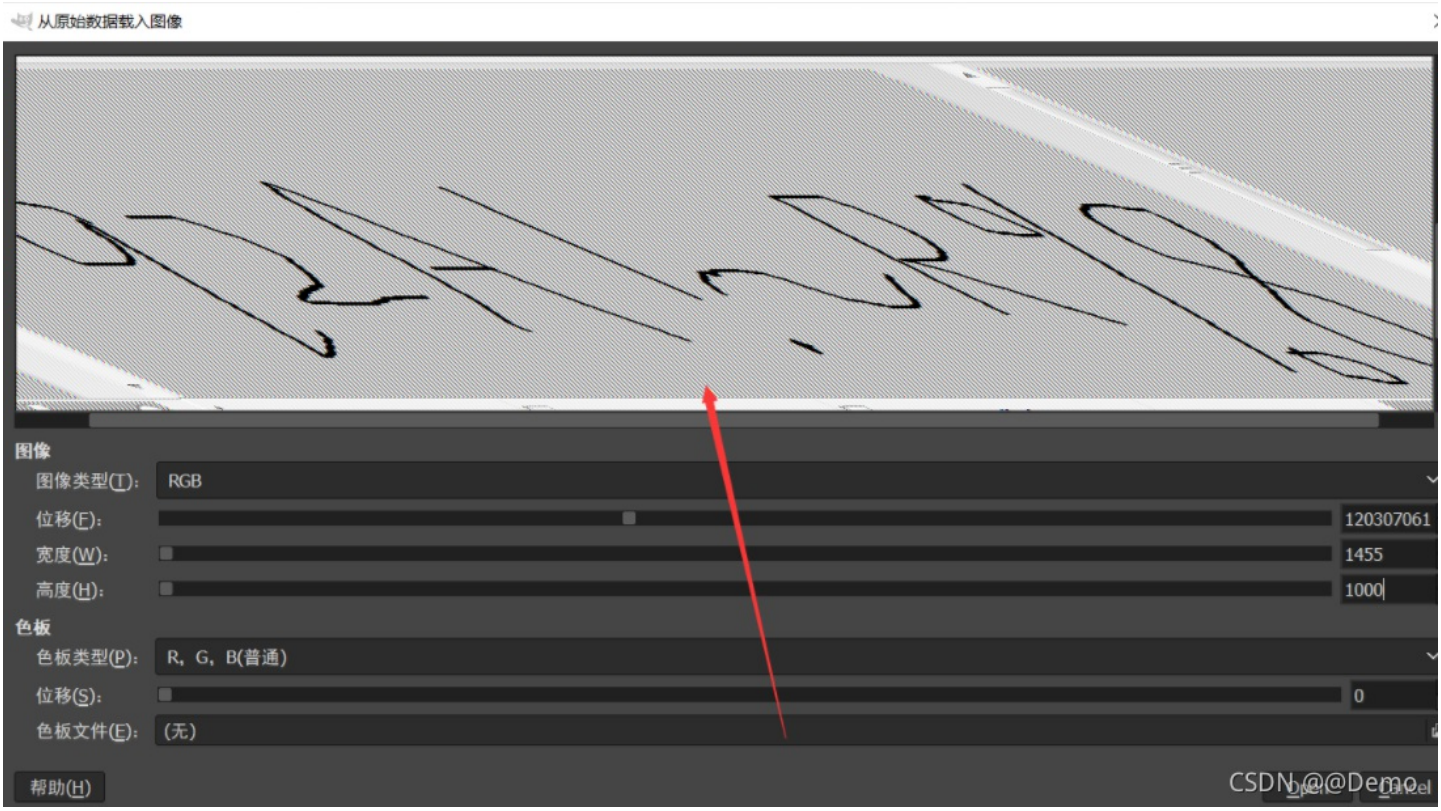
```
th1s_1s_th3_k3y
```

在看进程的时候有一个dumpit.exe,dump一下这个进程

```
volatility -f mspaint.raw --profile=Win7SP1x64 memdump -p 1064 -D ./
```

```
root@kali:~/volatility_2.6_lin64_standalone# volatility -f mspaint.raw --profile=Win7SP1x64 memdump -p 1064 -D
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing DumpIt.exe [  1064] to 1064.dmp
```

调整参数



得到压缩包的解压密码

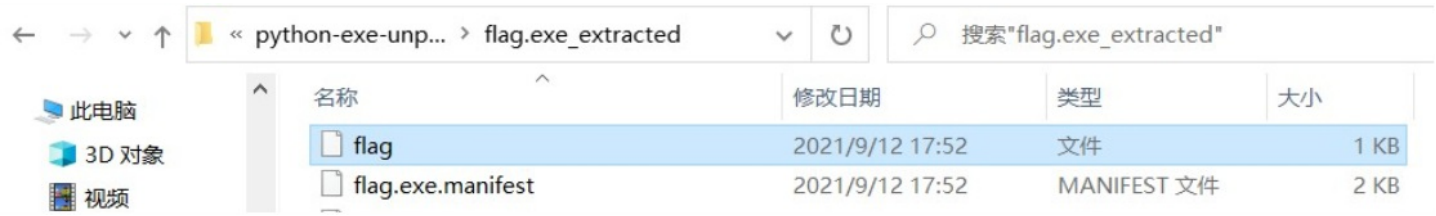
```
q2A!~R%8
```

后面是队里的re手个人秀

把flag.exe转成flag.pyc

```
D:\Safe\bin\python-exe-unpacker-master
λ python D:\Safe\bin\python-exe-unpacker-master\pyinstxtractor.py C:\Users\User\Desktop\flag.exe
D:\Safe\bin\python-exe-unpacker-master\pyinstxtractor.py:95: DeprecationWarning: the imp module is deprecated in favou
r of importlib; see the module's documentation for alternative uses
  import imp
[*] Processing C:\Users\User\Desktop\flag.exe
[*] Pyinstaller version: 2.1+
[*] Python version: 38
[*] Length of package: 7224898 bytes
[*] Found 69 files in CArchive
[*] Beginning extraction...please standby
[*] Found 206 files in PYZ archive
[*] Successfully extracted pyinstaller archive: C:\Users\User\Desktop\flag.exe

You can now use a python decompiler on the pyc files within the extracted directory
```


文件夹里出现



修改文件头为pyc

```
D:\Safe\bin\python-exe-unpacker-master\flag.exe_extracted
λ uncompyle6 flag.pyc > flag.py
Traceback (most recent call last):
  File "e:\python\python3.8\lib\site-packages\xdis\load.py", line 293, in load_module_from_file_object
    co = marshal.loads(bytecode)
ValueError: bad marshal data (unknown type code)
Traceback (most recent call last):
  File "e:\python\python3.8\lib\site-packages\xdis\load.py", line 293, in load_module_from_file_object
    co = marshal.loads(bytecode)
ValueError: bad marshal data (unknown type code)

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "e:\python\python3.8\lib\runpy.py", line 194, in _run_module_as_main
    return _run_code(code, main_globals, None,
  File "e:\python\python3.8\lib\runpy.py", line 87, in _run_code
    exec(code, run_globals)
  File "E:\Python\Python3.8\Scripts\uncompyle6.exe\__main__.py", line 7, in <module>
```



```
key = 'xxxxxxxxxxxxxx'
flag = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
data = ''
for i in range(0, len(flag)):
    data += hex(ord(flag[i]) ^ ord(key[(i % 15)]))[2:].zfill(2)
else:
    print(data.upper())
    data = '120450142403436844505O6E5E1E1C165D045E6B52113C5951006F091E4F4C0C54426A52466A165B0122'
```

这里的key15位 就是那张图片里的th1s_1s_th3_k3y

写出解密脚本

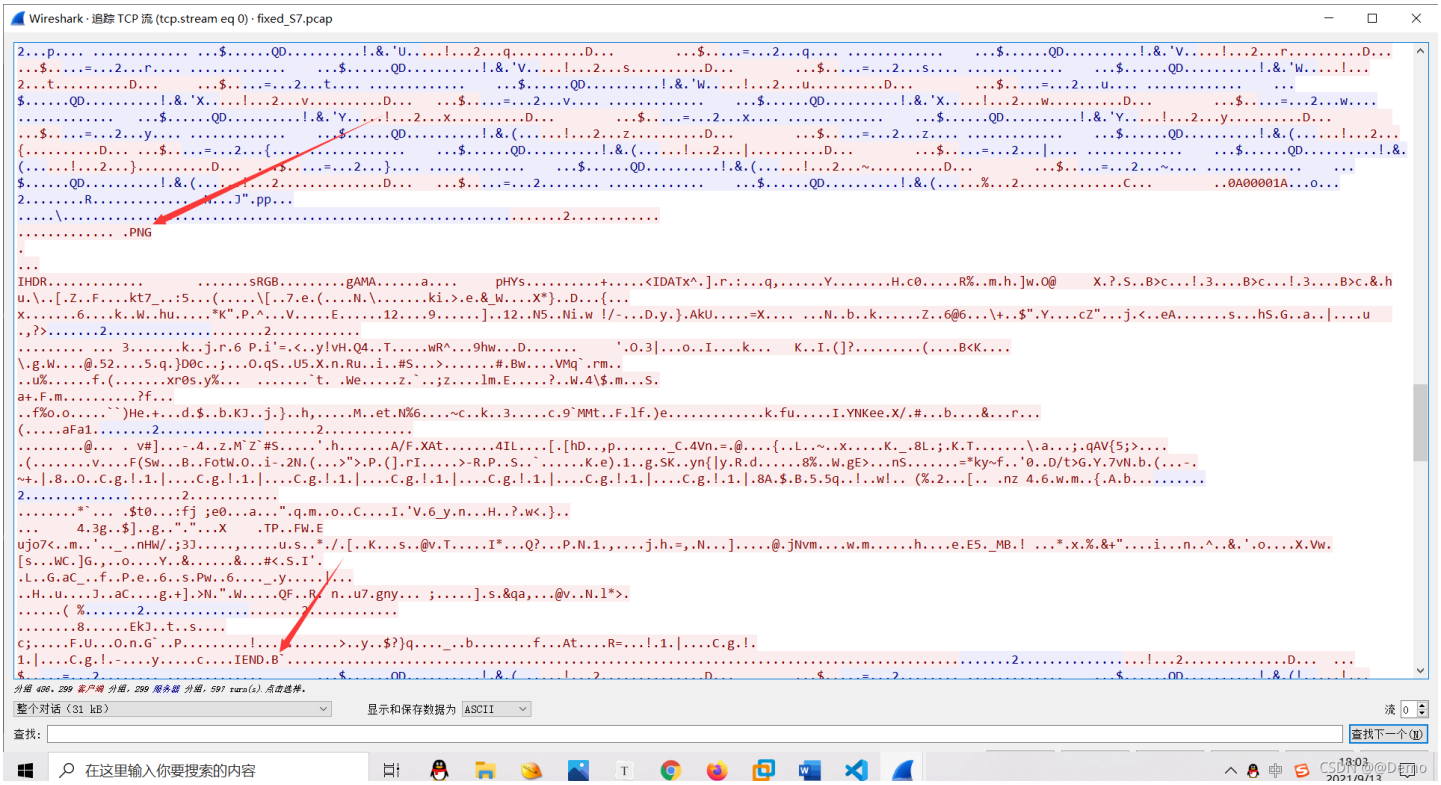
```
key = 'th1s_1s_th3_k3y'
flag = ''
data = '1204501424034368445050 6E5E1E1C165D045E6B52113C5951006F091E4F4C0C54426A52466A165B0122'
data_list = []
for i in range(0,len(data),2):
    data_list.append('0x' + data[i:i+2])
print(data_list)
for i in range(0,len(data_list)):
    flag += chr(int(data_list[i],16)^ord(key[(i % 15)]))

print(flag)
```

```
flag{20708c15-eb55-4cbc-930b-68de15c55b32}
```

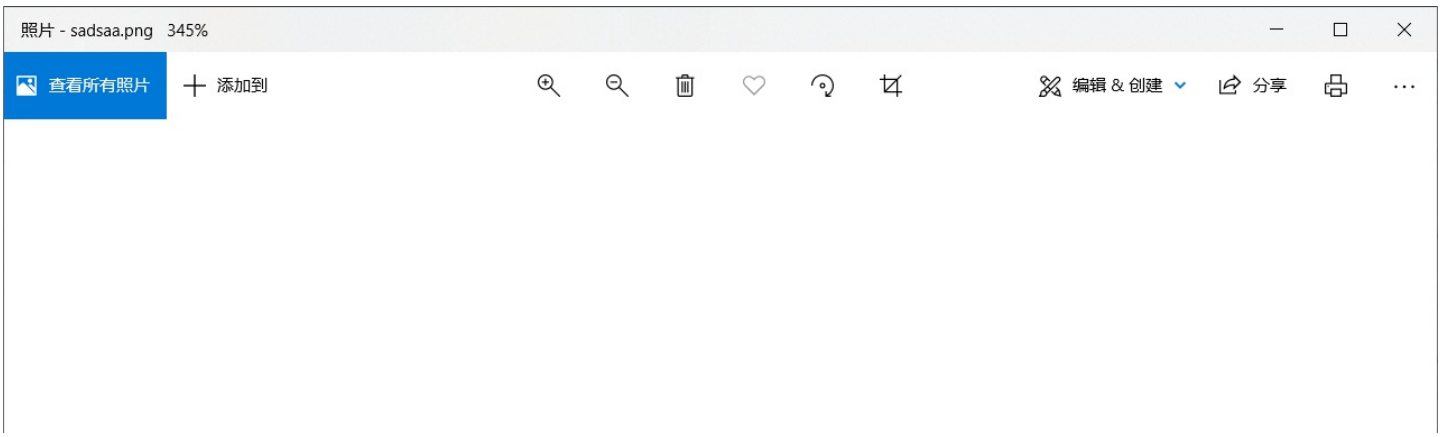
藏在s7里的秘密

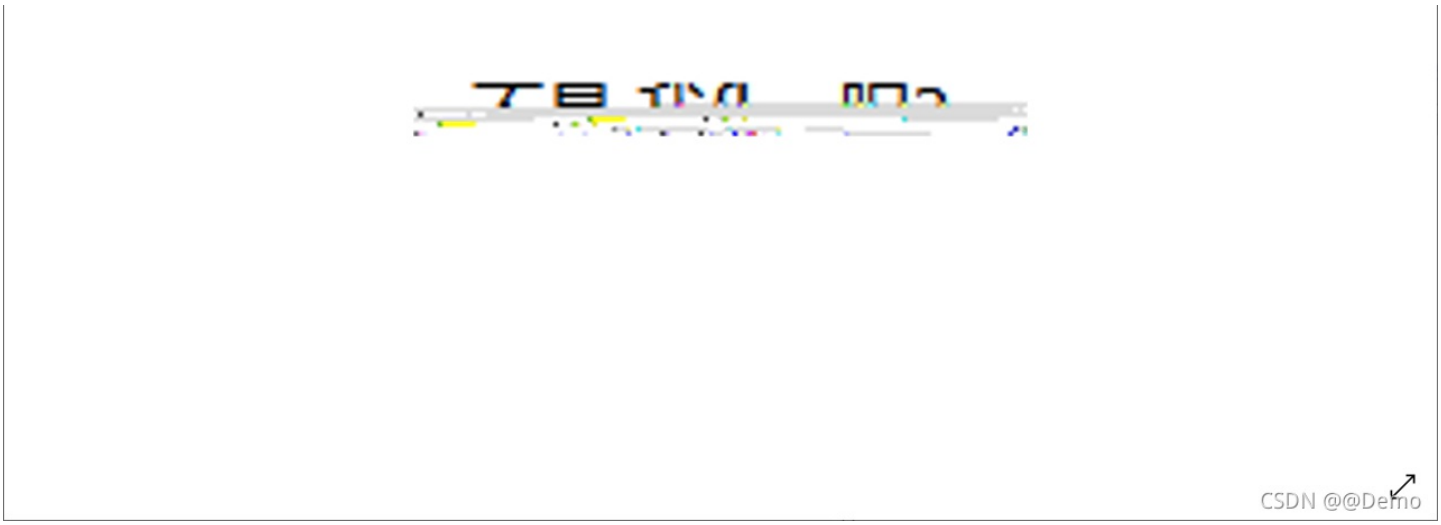
题目描述：某工厂的安全设备捕获了攻击者向PLC中写入恶意数据的数据包，你能分析出并找到其中隐藏的数据吗？

附件打开发现流量包损坏打不开，010打开发现文件头被修改了，直接pcapfix修复下就好。
打开后，追踪tcp流发现png图片



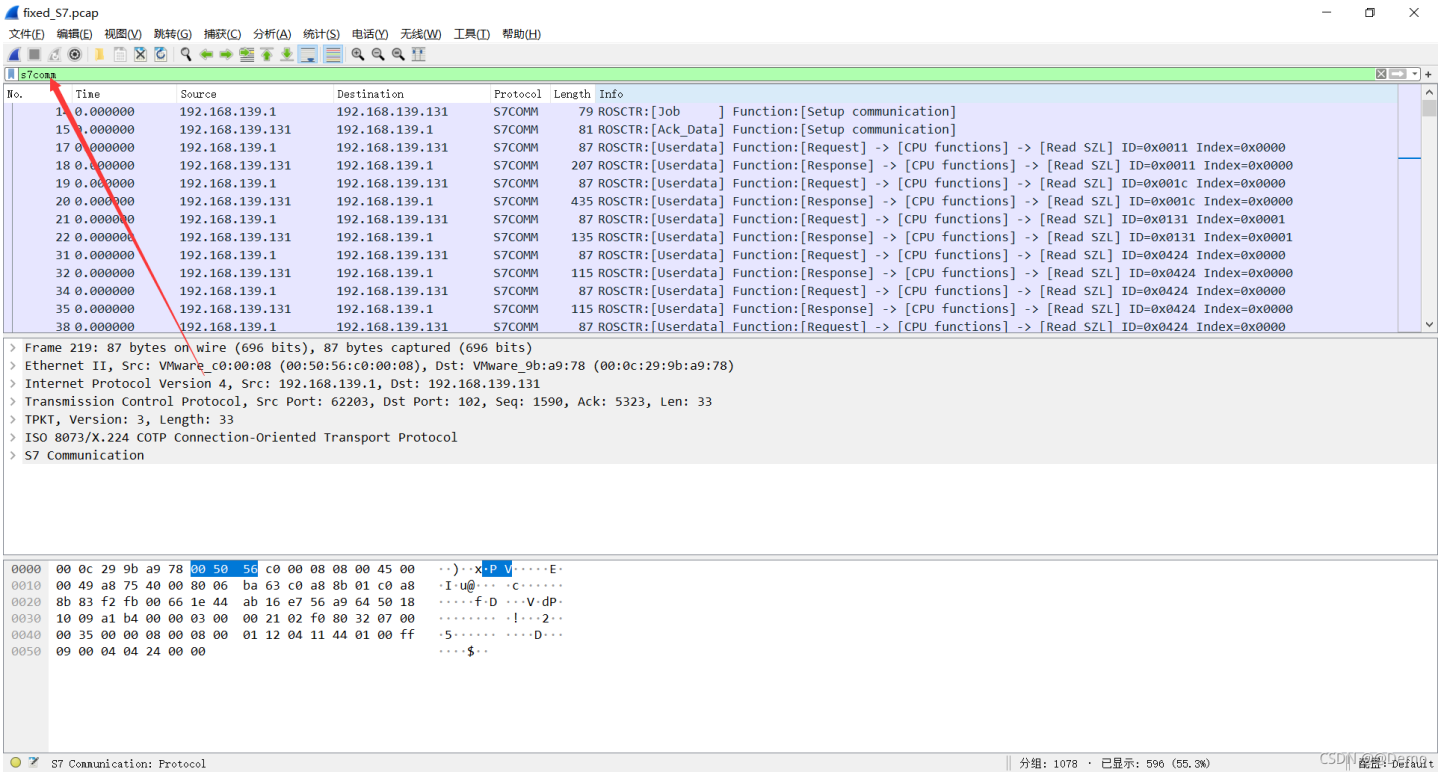
保存后发现只能看清楚一半的图



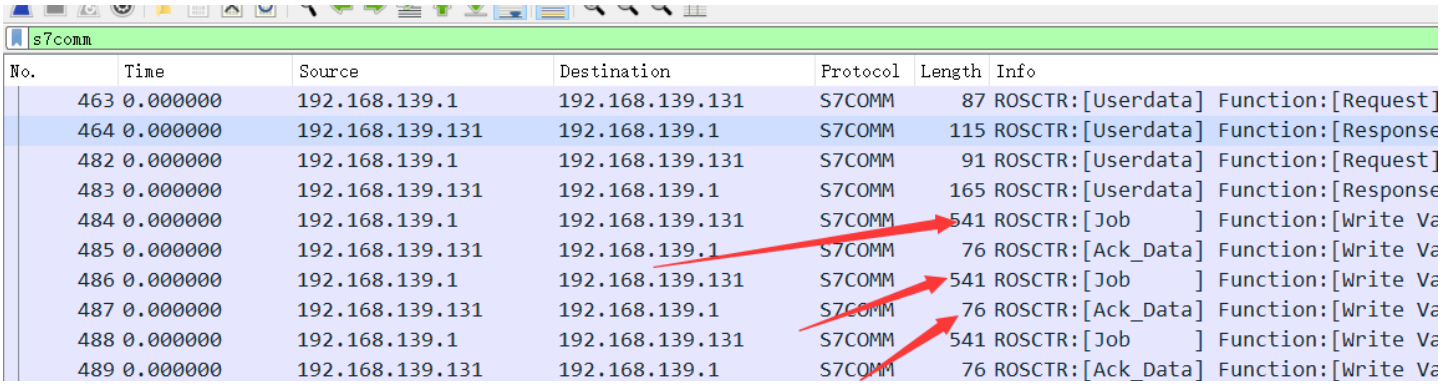


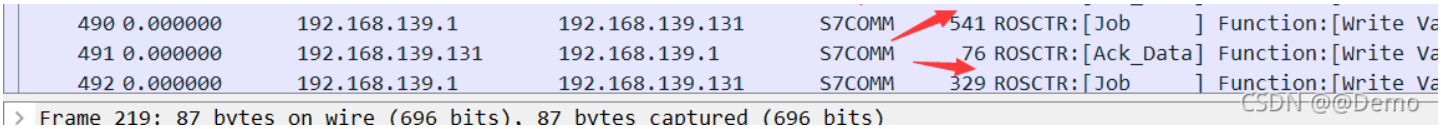
显然十六进制数据是不完整的，然后因为没咋做过流量包，到这思路也就停滞了。赛后看师傅的wp发现要看，这个流量包中出现最多的流s7comm，

先使用上面的过滤框，单独过滤这个协议



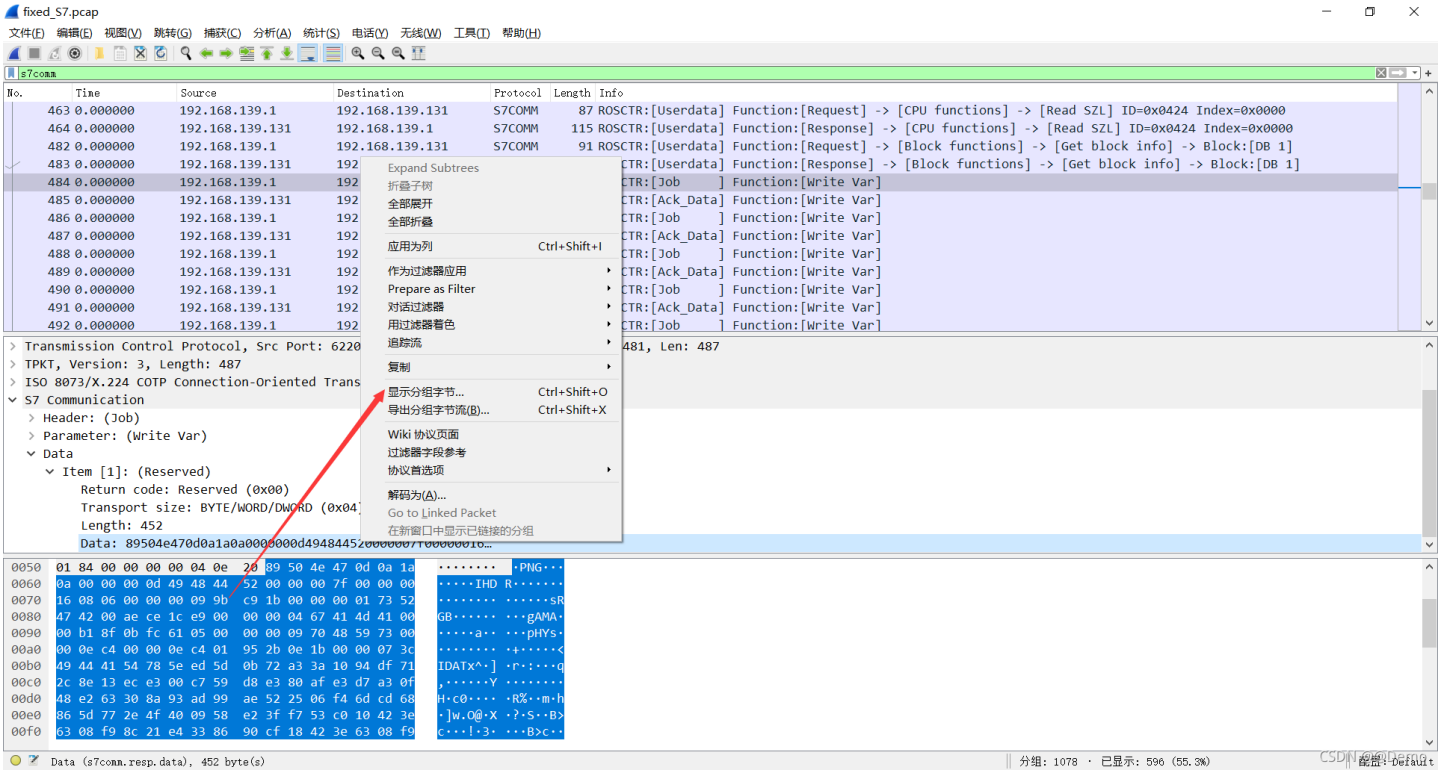
在查看这个流的时候发现长度有不一样的地方，看第一个长度541的包，发现存在png的文件头，最后一个长度329的包存在png的文件尾



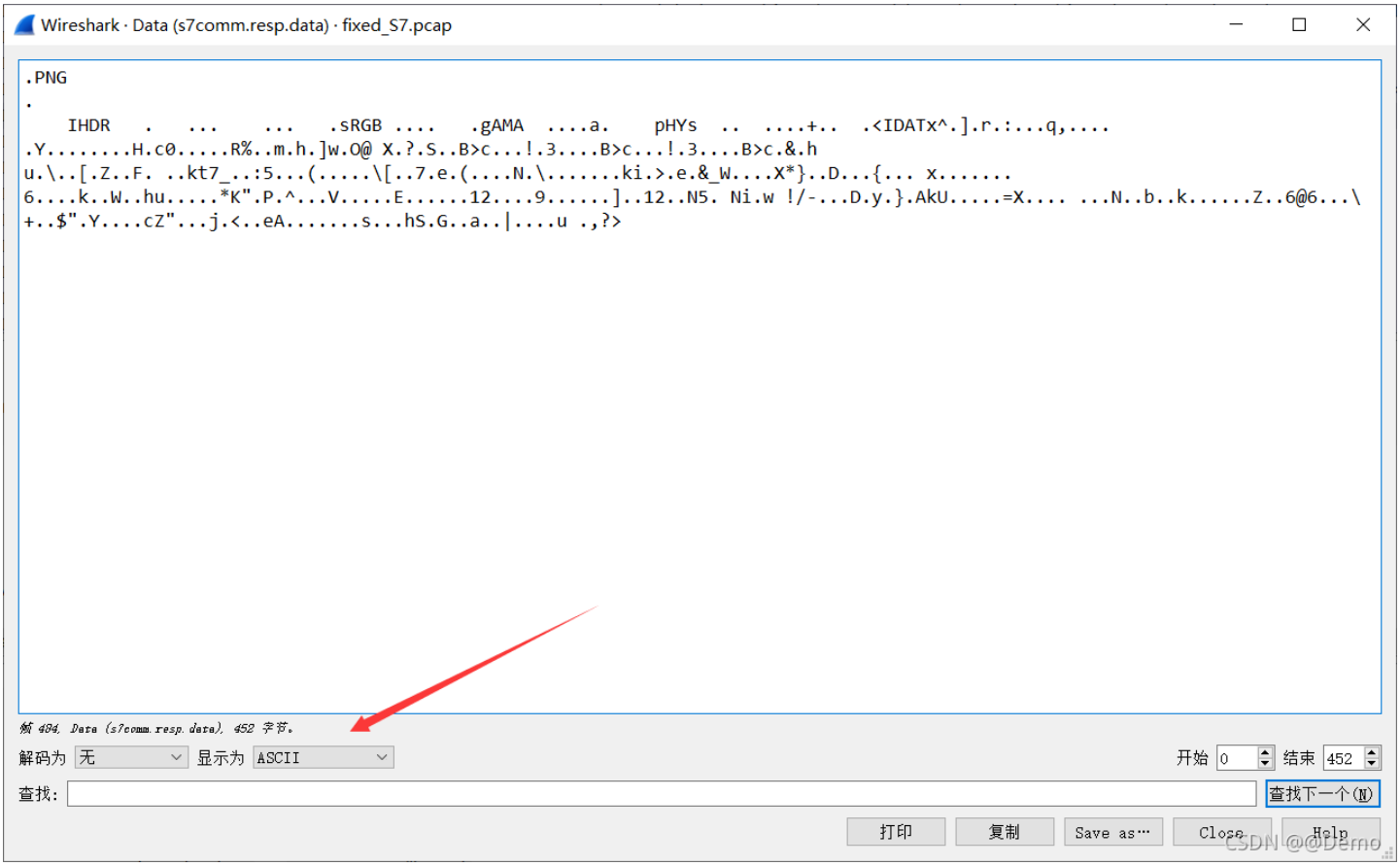


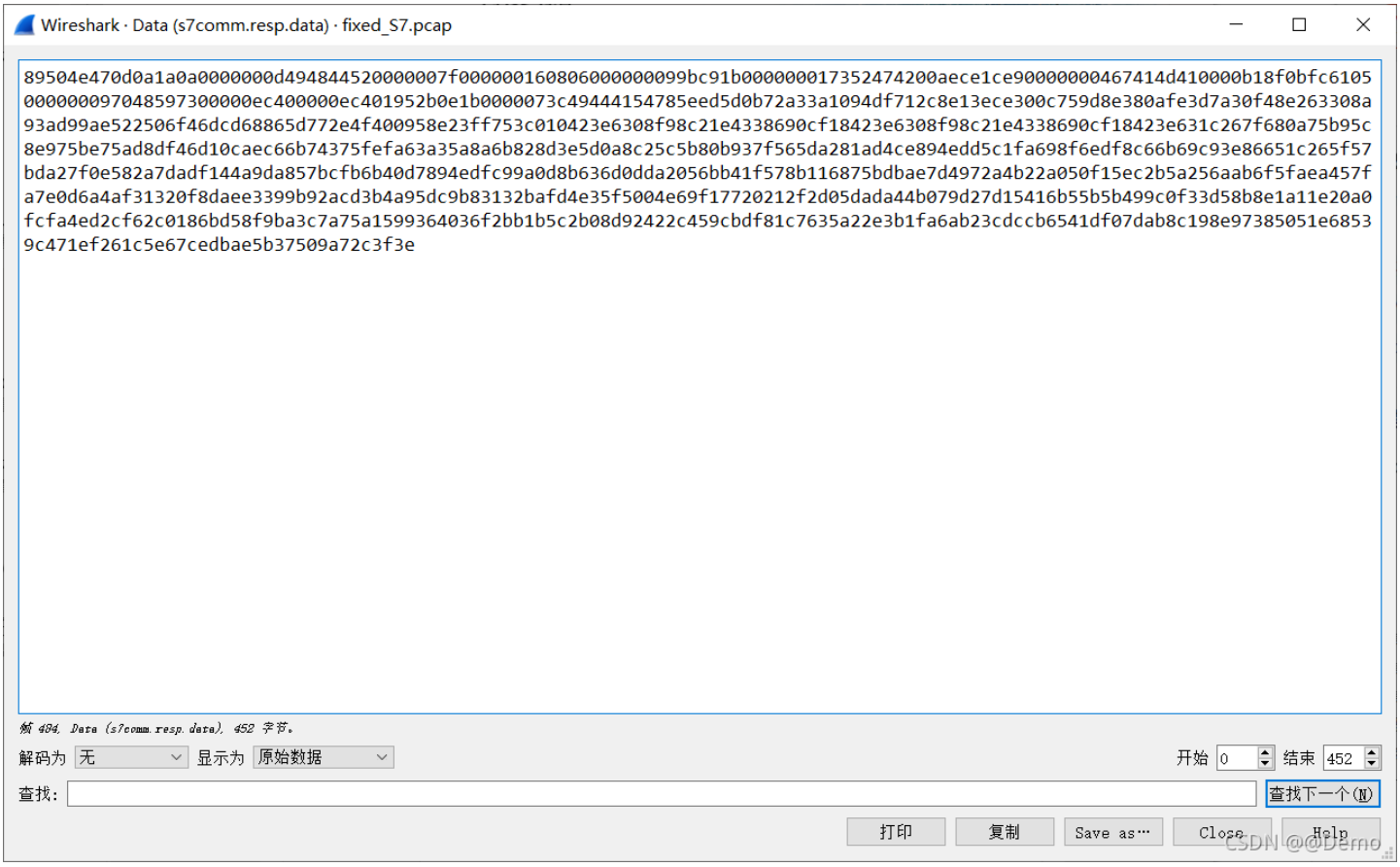
拿导出第一部分举例

选中data->显示分组字节

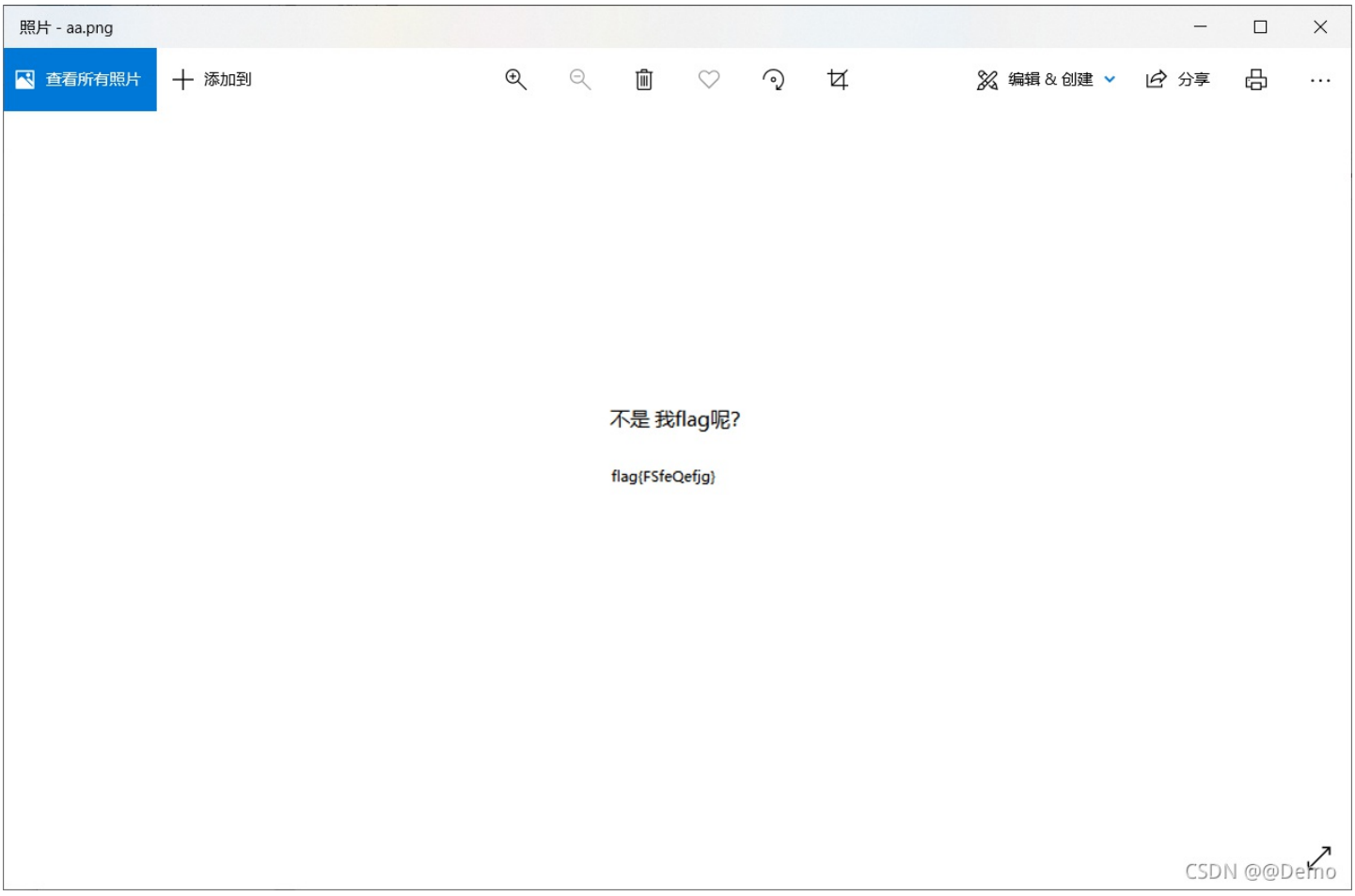


单击左下角改为原始数据





选中010,ctrl+shift+v保存即可。按照顺序依次把数据加进去即可。
保存后发现crc的值报错，说明宽高被修改了，修改高度获得flag

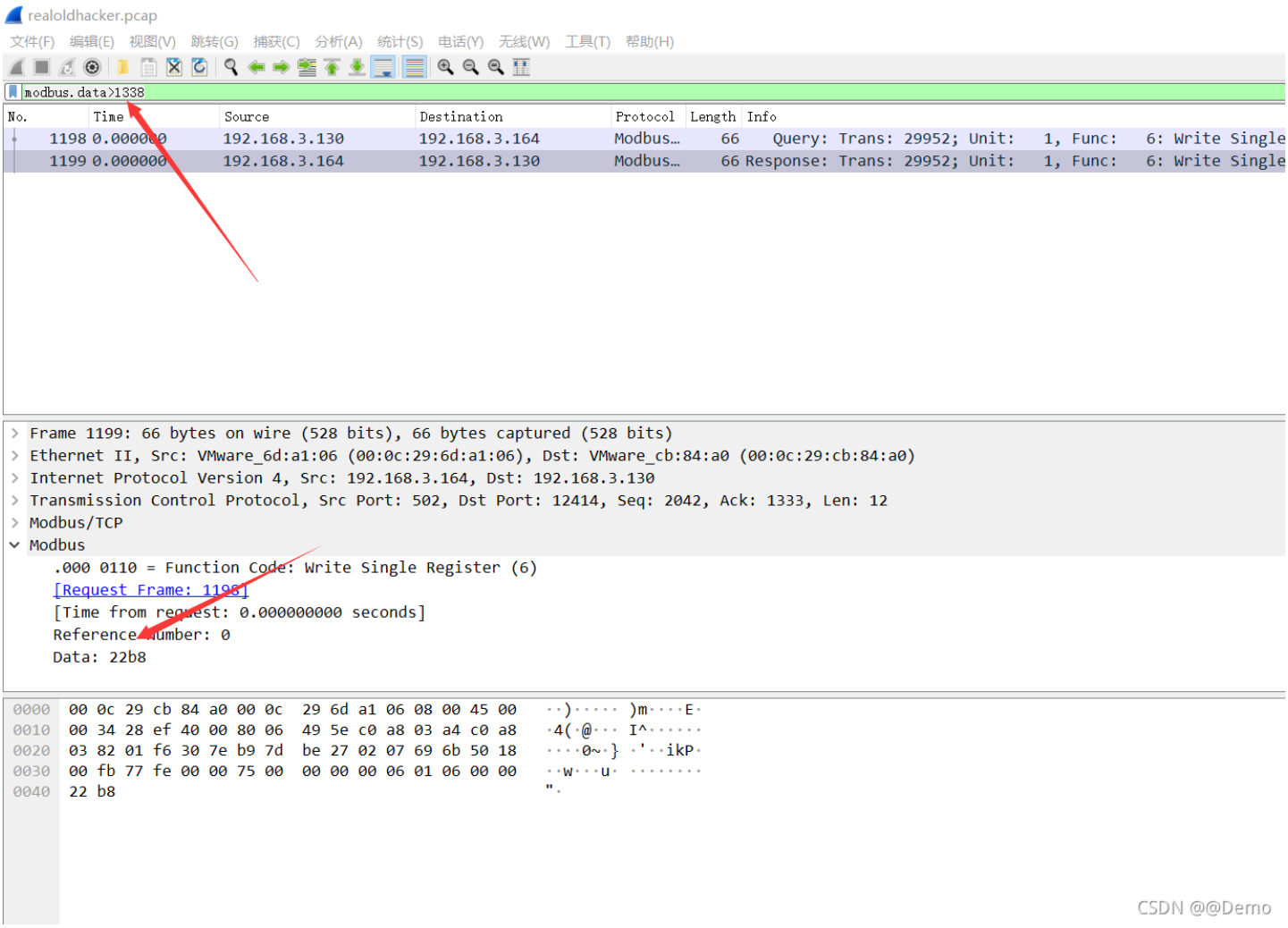


```
flag{FSfeQefjg}
```

老练的黑客

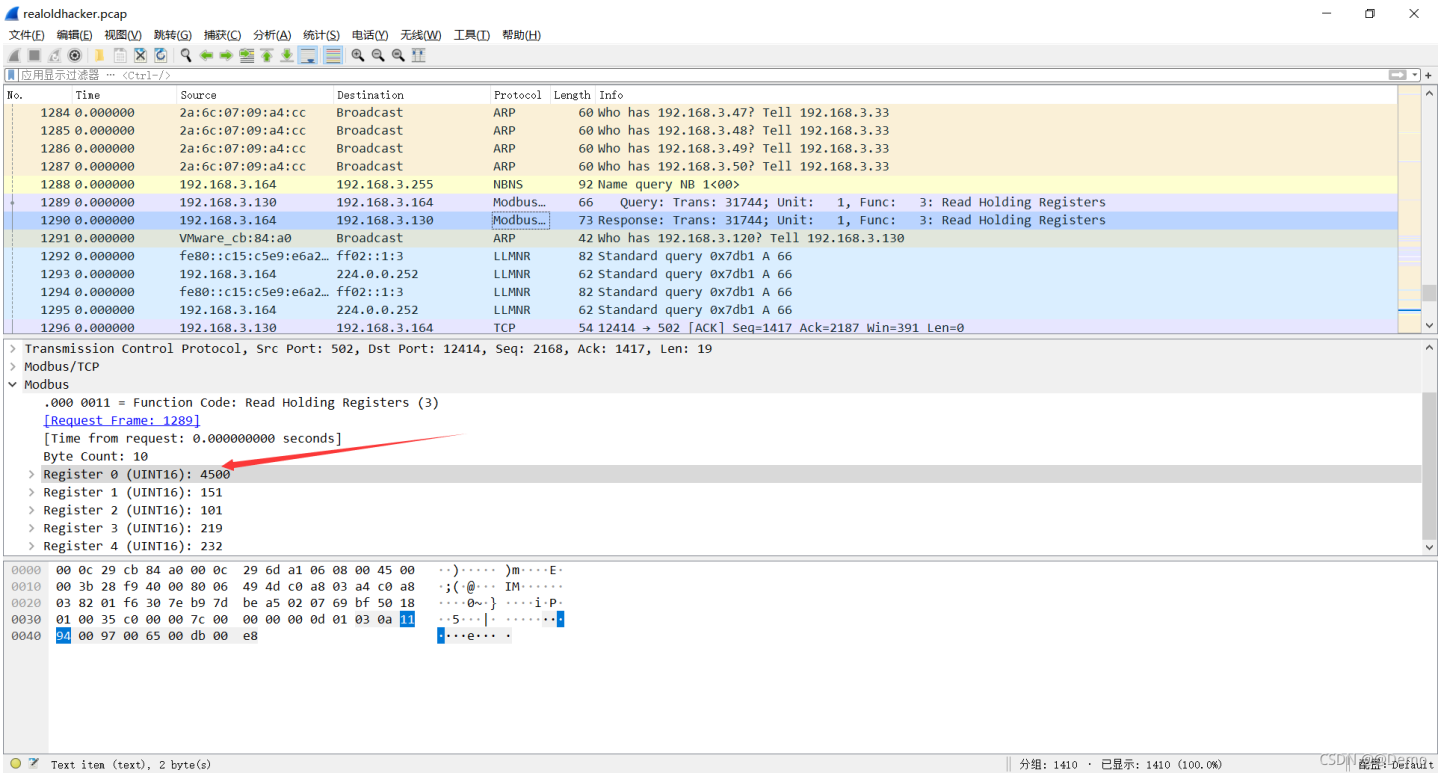
题目描述：一黑客成功入侵某核电站且获得了操作员站控制权，该操作员站可控制离心机的转速，当离心机的转速持续高于5000时将导致设备损坏，为了保护设备，操作员站检测到转速超过5000时会自动限制转速。但该黑客非常老练，他在修改了转速后，还欺骗了操作员站，使得操作员站读取到错误的转速数据。你能找到黑客修改后的转速值和操作员站读取到的错误转速值吗？（flag格式：flag{修改后转速的+读取的错误转速} 值用16进制表示
通过查看发现转速的数据位于data
5000的十六进制为1388，所以只要找到data大于1388的包查看数据即可
使用过滤器

```
modbus.data>1338
```



所以第一部分的flag为22b8。

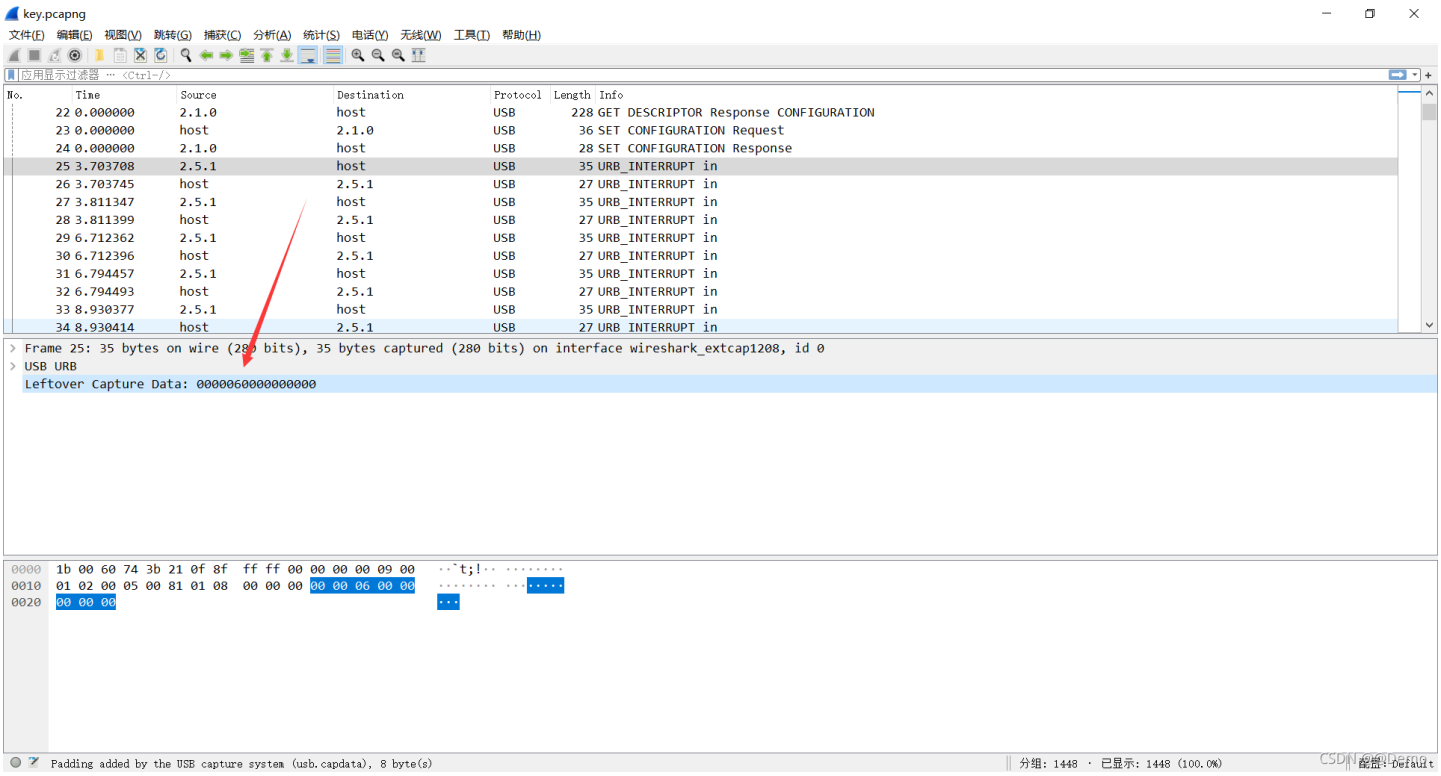
又说转速被修改了，往下翻倒数read的几个包，找到一个4500对应十六进制为1194,肯定是被修改的转速。



```
flag{22b81194}
```

baby-usb

usb流量找到有数据的包，发现是八字节的，可能是键盘流量。



那就找键盘流量的脚本梭一下。
用脚本直接跑没啥输出，所以先用tshark提取了一下

```
tshark -r key.pcapng -T fields -e usb.capdata | sed '/^\s*$/d' > fileaaa.txt
```

然后修改了下脚本有输出了

```
import os,sys

normalKeys = {
    "04":"a", "05":"b", "06":"c", "07":"d", "08":"e",
    "09":"f", "0a":"g", "0b":"h", "0c":"i", "0d":"j",
     "0e":"k", "0f":"l", "10":"m", "11":"n", "12":"o",
      "13":"p", "14":"q", "15":"r", "16":"s", "17":"t",
       "18":"u", "19":"v", "1a":"w", "1b":"x", "1c":"y",
        "1d":"z","1e":"1", "1f":"2", "20":"3", "21":"4",
         "22":"5", "23":"6","24":"7","25":"8","26":"9",
         "27":"0","28":"<RET>","29":"<ESC>","2a":"<DEL>", "2b":"\t",
         "2c":"<SPACE>","2d":"-","2e":"=","2f":"[","30":"]","31":"\\",
         "32":"<NON>","33":";","34":"'","35":"<GA>","36":",","37":".",
         "38":"/","39":"<CAP>","3a":"<F1>","3b":"<F2>", "3c":"<F3>","3d":"<F4>",
         "3e":"<F5>","3f":"<F6>","40":"<F7>","41":"<F8>","42":"<F9>","43":"<F10>",
         "44":"<F11>","45":"<F12>"}
shiftKeys = {
    "04":"A", "05":"B", "06":"C", "07":"D", "08":"E",
     "09":"F", "0a":"G", "0b":"H", "0c":"I", "0d":"J",
      "0e":"K", "0f":"L", "10":"M", "11":"N", "12":"O",
       "13":"P", "14":"Q", "15":"R", "16":"S", "17":"T",
        "18":"U", "19":"V", "1a":"W", "1b":"X", "1c":"Y",
         "1d":"Z","1e":"!", "1f":"@", "20":"#", "21":"$",
```

```
          "22":"%", "23":"^","24":"&","25":"*","26":"(","27":")",
          "28":"<RET>","29":"<ESC>","2a":"<DEL>", "2b":"\t","2c":"<SPACE>",
          "2d":"_","2e":"+","2f":"{","30":"}","31":"|","32":"<NON>","33":"\"",
          "34":":","35":"<GA>","36":"<","37":">","38":"?","39":"<CAP>","3a":"<F1>",
          "3b":"<F2>", "3c":"<F3>","3d":"<F4>","3e":"<F5>","3f":"<F6>","40":"<F7>",
          "41":"<F8>","42":"<F9>","43":"<F10>","44":"<F11>","45":"<F12>"}

#pcapFilePath = sys.argv[1]
#os.system("tshark -r "+pcapFilePath+" -T fields -e usb.capdata | sed '/^\s*$/d' > out.txt")

output = []
keys = open('out.txt')
for line in keys:
    line = ''.join(line[i:i+2]+':' for i in range(0,len(line)-1,2)).strip(':')
    try:
        if line[0]!='0' or (line[1]!='0' and line[1]!='2') or line[3]!='0' or line[4]!='0' or line[9]!='0' or li
ne[10]!='0' or line[12]!='0' or line[13]!='0' or line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0
' or line[21]!='0' or line[22]!='0' or line[6:8]=="00":
            continue
        if line[6:8] in normalKeys.keys():
            output += [[normalKeys[line[6:8]]],[shiftKeys[line[6:8]]]][line[1]=='2']
        else:
            output += ['[unknown]']
    except:
        pass

keys.close()

flag=0
#print("".join(output))
for i in range(len(output)):
    try:
        a=output.index('<DEL>')
        del output[a]
        del output[a-1]
    except:
        pass

for i in range(len(output)):
    try:
        if output[i]=="<CAP>":
            flag+=1
            output.pop(i)
            if flag==2:
                flag=0
        if flag!=0:
            output[i]=output[i].upper()
    except:
        pass

print ('output :' + "".join(output))
os.system("rm -rf out.txt"
```

```
output :congratulationsonfindingmebutiwillnottellyouwherethepasswordofworddocumentisgoandfinditagain
```

到这其实就没什么思路了，尝试了各种东西，赛后看发现方向错了。
借用盖乐希师傅的脚本

```
normalKeys = {
```

```
        "04":"a", "05":"b", "06":"c", "07":"d", "08":"e",
        "09":"f", "0a":"g", "0b":"h", "0c":"i", "0d":"j",
         "0e":"k", "0f":"l", "10":"m", "11":"n", "12":"o",
          "13":"p", "14":"q", "15":"r", "16":"s", "17":"t",
           "18":"u", "19":"v", "1a":"w", "1b":"x", "1c":"y",
            "1d":"z","1e":"1", "1f":"2", "20":"3", "21":"4",
             "22":"5", "23":"6","24":"7","25":"8","26":"9",
             "27":"0","28":"<RET>","29":"<ESC>","2a":"<DEL>", "2b":"\t",
             "2c":"<SPACE>","2d":"-","2e":"=","2f":"[","30":"]","31":"\\",
             "32":"<NON>","33":";","34":"'","35":"<GA>","36":",","37":".",
             "38":"/","39":"<CAP>","3a":"<F1>","3b":"<F2>", "3c":"<F3>","3d":"<F4>",
             "3e":"<F5>","3f":"<F6>","40":"<F7>","41":"<F8>","42":"<F9>","43":"<F10>",
             "44":"<F11>","45":"<F12>"}
shiftKeys = {
        "04":"A", "05":"B", "06":"C", "07":"D", "08":"E",
         "09":"F", "0a":"G", "0b":"H", "0c":"I", "0d":"J",
          "0e":"K", "0f":"L", "10":"M", "11":"N", "12":"O",
           "13":"P", "14":"Q", "15":"R", "16":"S", "17":"T",
            "18":"U", "19":"V", "1a":"W", "1b":"X", "1c":"Y",
             "1d":"Z","1e":"!", "1f":"@", "20":"#", "21":"$",
              "22":"%", "23":"^","24":"&","25":"*","26":"(","27":")",
              "28":"<RET>","29":"<ESC>","2a":"<DEL>", "2b":"\t","2c":"<SPACE>",
              "2d":"_","2e":"+","2f":"{","30":"}","31":"|","32":"<NON>","33":"\"",
              "34":":","35":"<GA>","36":"<","37":">","38":"?","39":"<CAP>","3a":"<F1>",
              "3b":"<F2>", "3c":"<F3>","3d":"<F4>","3e":"<F5>","3f":"<F6>","40":"<F7>",
              "41":"<F8>","42":"<F9>","43":"<F10>","44":"<F11>","45":"<F12>"}
output = []
keys = open('filebbb.txt')
for line in keys:
    try:
        if line[0]!='0' or (line[1]!='0' and line[1]!='2') or line[3]!='0' or line[4]!='0' or line[9]!='0' or li
ne[10]!='0' or line[12]!='0' or line[13]!='0' or line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0
' or line[21]!='0' or line[22]!='0' or line[6:8]=="00":
            continue
        if line[6:8] in normalKeys.keys():
            output += [[normalKeys[line[6:8]]],[shiftKeys[line[6:8]]]][line[1]=='2']
        else:
            output += ['[unknown]']
    except:
        pass

keys.close()

flag=0
print("".join(output))
for i in range(len(output)):
    try:
        a=output.index('<DEL>')
        del output[a]
        del output[a-1]
    except:
        pass

for i in range(len(output)):
    try:
        if output[i]=="<CAP>":
            flag+=1
            output.pop(i)
        if flag==2:
```

```
            flag=0
        if flag!=0:
            output[i]=output[i].upper()
    except:
        pass

print ('output :' + "".join(output))
```

```
ct<DEL>onh<DEL>gratue<DEL>latioke<DEL><DEL>nsonfiny<DEL>dingmebutiwii<DEL>llns<DEL>ottellyouwheretqa<DEL><DEL>he
pz<DEL>asswordws<DEL><DEL>ox<DEL>fwe<DEL>od<DEL>rddoc<DEL>cumentisgoarfv<DEL><DEL><DEL>ndfinditagain
output :congratulationsonfindingmebutiwillnottellyouwherethepasswordofworddocumentisgoandfinditagain
```

每个前面的字符就是~~key qazwsxedcrfv~~

```
flag{685b42b0-da3d-47f4-a76c-0f3d07ea962a}
```