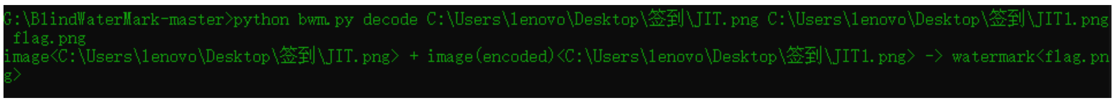
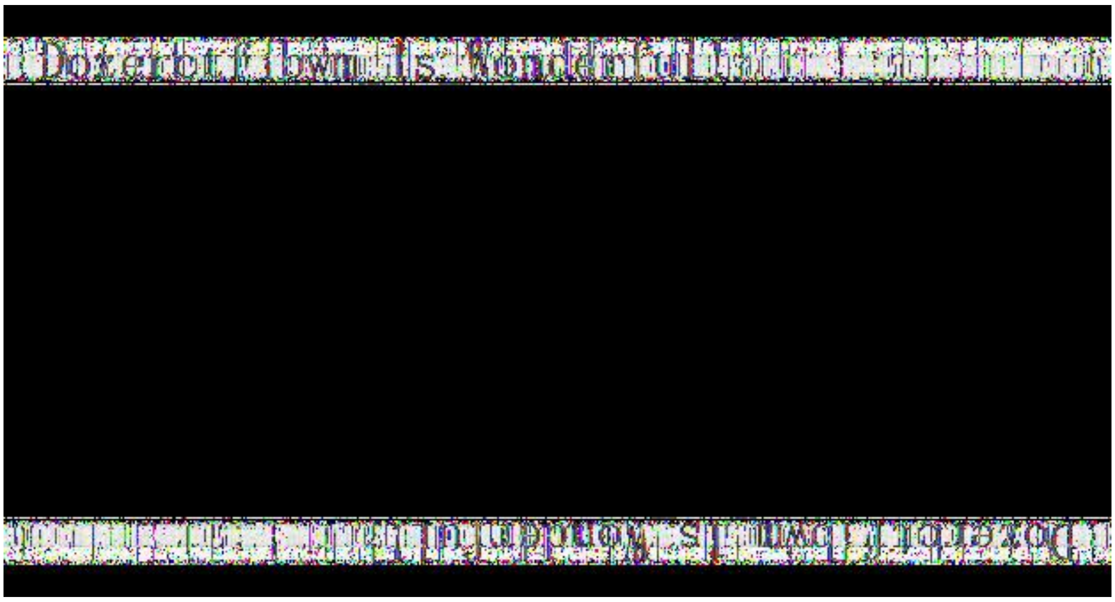
1. 签到（难度：简单）

盲水印，使用 BlindWaterMark（项目地址 https://github.com/chishaxie/BlindWaterMark）工具获取水印信息。



```
G:\BlindWaterMark-master>python bwm.py decode C:\Users\lenovo\Desktop\签到\JIT.png C:\Users\lenovo\Desktop\签到\JIT1.png
 flag.png
image<C:\Users\lenovo\Desktop\签到\JIT.png> + image(encoded)<C:\Users\lenovo\Desktop\签到\JIT1.png> -> watermark<flag.pn
g>
```

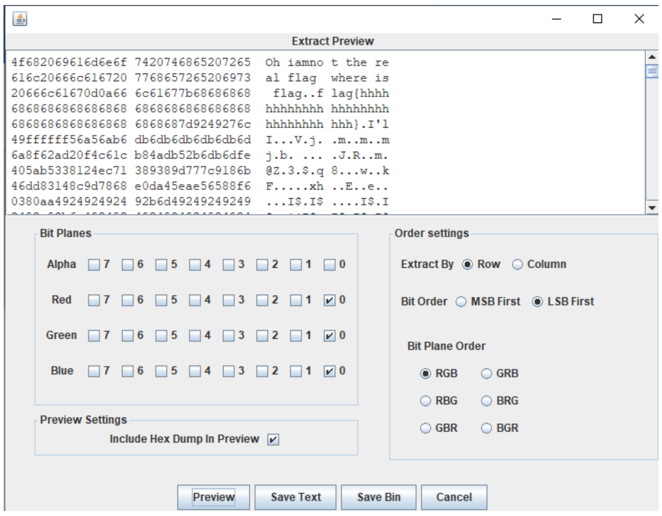
命令：python bwm.py decode <image> <image(encoded)> <watermark>



flag：Dozerctf{bwm_1s_Wonderful!}

2. 夏日计划（难度：中等）

LSB 发现是假 flag



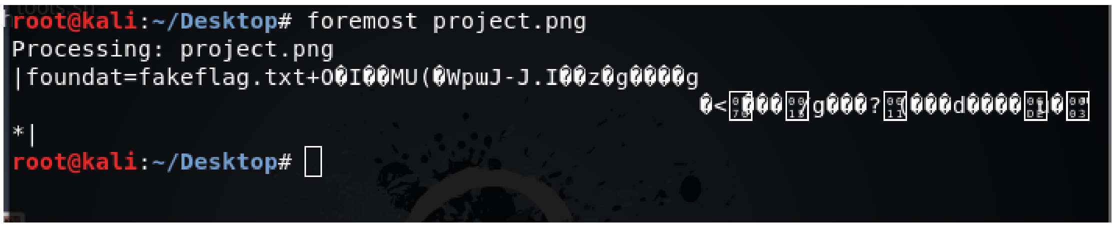
```
4f682069616d6e6f 7420746865207265  Oh iamno t the re
616c20666c616720 7768657265206973  al flag  where is
20666c61670d0a66 6c61677b68686868   flag..f lag{hhhh
6868686868686868 6868686868686868  hhhhhhhh hhhhhhhh
6868686868686868 6868687d9249276c  hhhhhhhh hhh}.I'l
49ffffff56a56ab6 db6db6db6db6db6d  I...V.j. .m..m..m
6a8f62ad20f4c61c b84adb52b6db6dfe  j.b. ... .J.R..m.
405ab5338124ec71 389389d777c9186b  @Z.3.$.q 8...w..k
46dd83148c9d7868 e0da45eae56588f6  F.....xh ..E..e..
0380aa4924924924 92b6d49249249249  ...I$.I$ ....I$.I
```

Bit Planes: Alpha: none; Red: [✓ 0]; Green: [✓ 0]; Blue: [✓ 0]

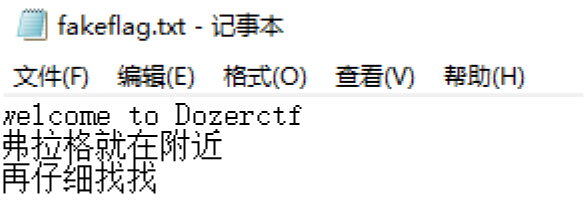
Preview Settings: Include Hex Dump In Preview [✓]

Order settings: Extract By [✓ Row] Column; Bit Order MSB First [✓ LSB First]; Bit Plane Order [✓ RGB] GRB RBG BRG GBR BGR

foremost 得到假 flag



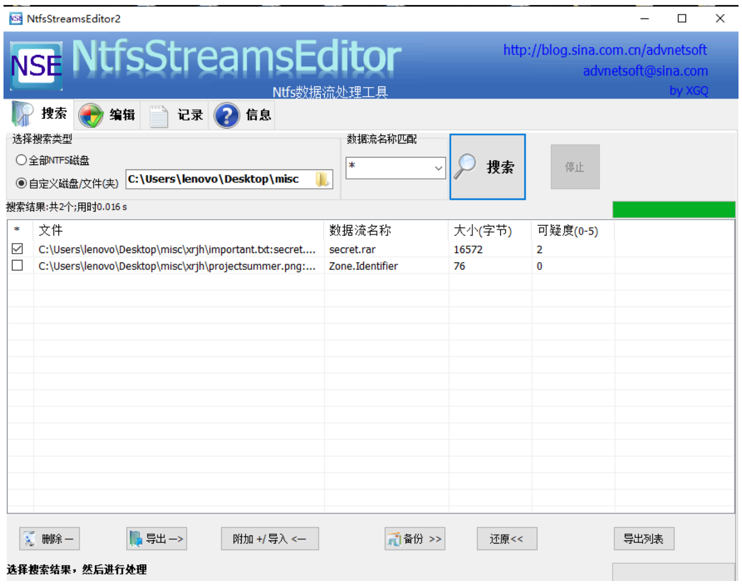
```
root@kali:~/Desktop# foremost project.png
Processing: project.png
|foundat=fakeflag.txt+O�I��MU(�WpwJ-J.I��z�g����g
                                                            �<���g���?���d�����
*|
root@kali:~/Desktop# 
```



fakeflag.txt - 记事本

```
welcome to Dozerctf
弗拉格就在附近
再仔细找找
```

取出 txt 中隐藏的 NTFS 数据流，将取出的文件合成一个文件。



自定义磁盘/文件(夹) C:\Users\lenovo\Desktop\misc

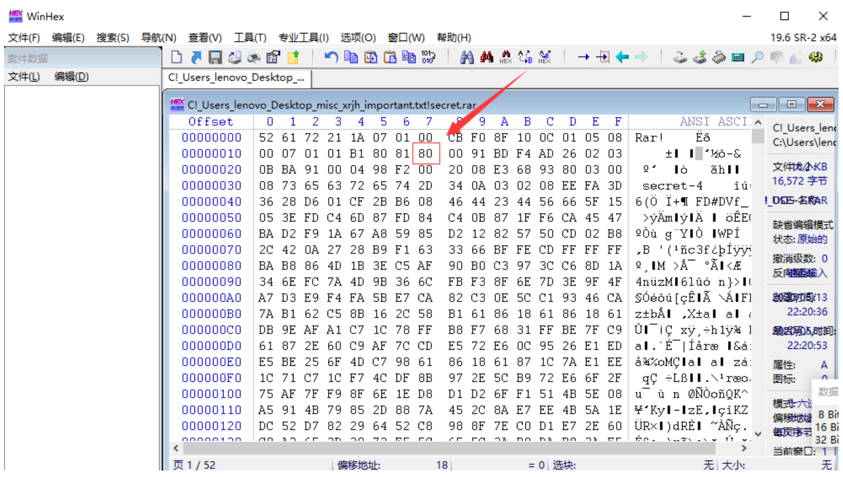
数据流名称匹配: *

搜索结果:共2个;用时0.016 s

*	文件	数据流名称	大小(字节)	可疑度(0-5)
☑	C:\Users\lenovo\Desktop\misc\xrjh\important.txt:secret....	secret.rar	16572	2
☐	C:\Users\lenovo\Desktop\misc\xrjh\projectsummer.png:...	Zone.Identifier	76	0

选择搜索结果，然后进行处理

导出 rar，伪加密将 4 改 0



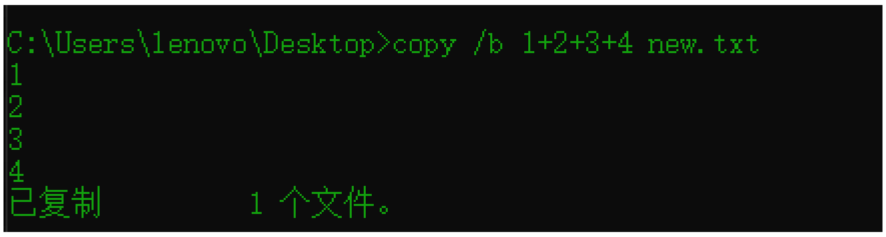
WinHex
文件(F) 编辑(E) 搜索(S) 导航(N) 查看(V) 工具(T) 专业工具(I) 选项(O) 窗口(W) 帮助(H)
19.6 SR-2 x64
案件数据
C!_Users_lenovo_Desktop_...

C!_Users_lenovo_Desktop_misc_xrjh_important.txt!secret.rar

```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F   ANSI ASCII
00000000   52 61 72 21 1A 07 01 00  CB F0 8F 10 0C 01 05 08   Rar!     Ëð
00000010   00 07 01 01 B1 80 81 80  00 91 BD F4 AD 26 02 03      ±     ´½ô-&
00000020   0B BA 91 00 04 98 F2 00  20 08 E3 68 93 80 03 00    º´   ò   ãh
00000030   08 73 65 63 72 65 74 2D  34 0A 03 02 08 EE FA 3D    secret-4    îú=
00000040   36 28 D6 01 CF 2B B6 08  46 44 23 44 56 66 5F 15   6(Ö Ï+¶ FD#DVf_
00000050   05 3E FD C4 6D 87 FD 84  C4 0B 87 1F F6 CA 45 47    >ýÄm ý Ä  öÊEG
00000060   BA D2 F9 1A 67 A8 59 85  D2 12 82 57 50 CD 02 B8   ºÒù g¨Y Ò  WPÍ ¸
00000070   2C 42 0A 27 28 B9 F1 63  33 66 BF FE CD FF FF FF   ,B '(¹ñc3f¿þÍÿÿÿ
00000080   BA B8 86 4D 1B 3E C5 AF  90 B0 C3 97 3C C6 8D 1A   º¸ M >Å¯ °Ã <Æ
00000090   34 6E FC 7A 4D 9B 36 6C  FB F3 8F 6E 7D 3E 9F 4F   4nüzM 6lûó n}> O
000000A0   A7 D3 E9 F4 FA 5B E7 CA  82 C3 0E 5C C1 93 46 CA   §Óéôú[çÊ Ã \Á FÊ
000000B0   7A B1 62 C5 8B 16 2C 58  B1 61 86 18 61 86 18 61   z±bÅ  ,X±a  a  a
000000C0   DB 9E AF A1 C7 1C 78 FF  B8 F7 68 31 FF BE 7F C9   Û ¯¡Ç xÿ¸÷h1ÿ¾ É
000000D0   61 87 2E 60 C9 AF 7C CD  E5 72 E6 0C 95 26 E1 ED   a .`É¯|Íåræ  &áí
000000E0   E5 BE 25 6F 4D C7 98 61  86 18 61 87 1C 7A E1 EE   å¾%oMÇ a  a  záî
000000F0   1C 71 C7 1C F7 4C DF 8B  97 2E 5C B9 72 E6 6F 2F    qÇ ÷Lß  .\¹ræo/
00000100   75 AF 7F F9 8F 6E 1E D8  D1 D2 6F F1 51 4B 5E 08   u¯ ù n ØÑÒoñQK^
00000110   A5 91 4B 79 85 2D 88 7A  45 2C 8A E7 EE 4B 5A 1E   ¥´Ky - zE, çîKZ
00000120   DC 52 D7 82 29 64 52 C8  98 8F 7E C0 D1 E7 2E 60   ÜR× )dRÈ  ~ÀÑç.`
```

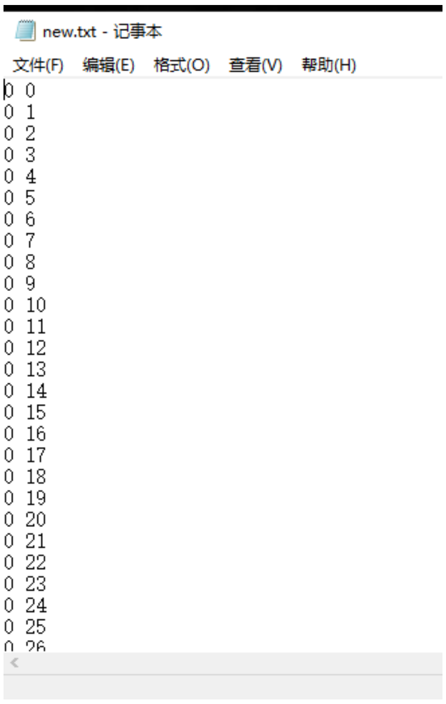
页 1 / 52　偏移地址: 18　= 0　选块: 无　大小:

将里面文件 4 合 1

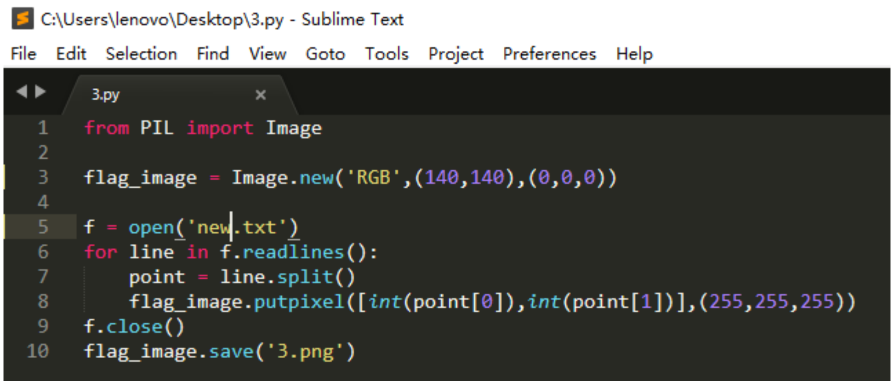


```
C:\Users\lenovo\Desktop>copy /b 1+2+3+4 new.txt
1
2
3
4
已复制         1 个文件。
```

将文档里的坐标转化为图片

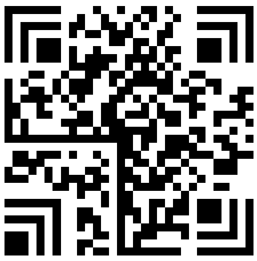


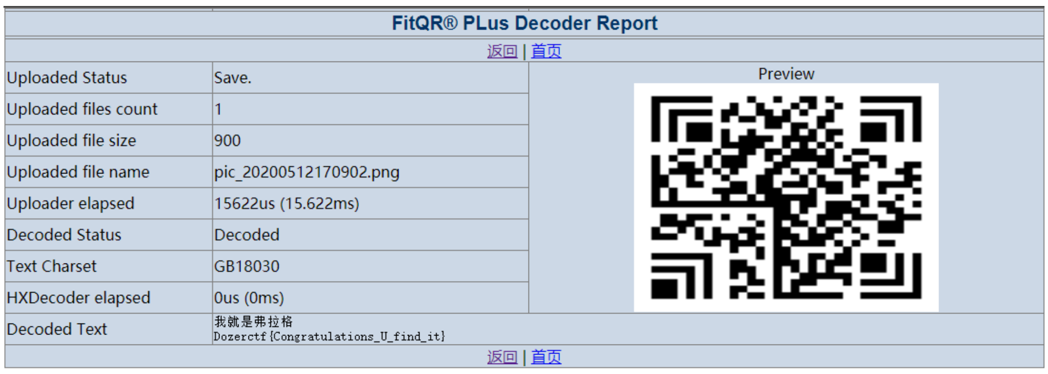
```
0 0
0 1
0 2
0 3
0 4
0 5
0 6
0 7
0 8
0 9
0 10
0 11
0 12
0 13
0 14
0 15
0 16
0 17
0 18
0 19
0 20
0 21
0 22
0 23
0 24
0 25
0 26
```


C:\Users\lenovo\Desktop\3.py - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
3.py

```python
from PIL import Image

flag_image = Image.new('RGB',(140,140),(0,0,0))

f = open('new.txt')
for line in f.readlines():
    point = line.split()
    flag_image.putpixel([int(point[0]),int(point[1])],(255,255,255))
f.close()
flag_image.save('3.png')
```

最终得到一个汉信码扫码可得 flag





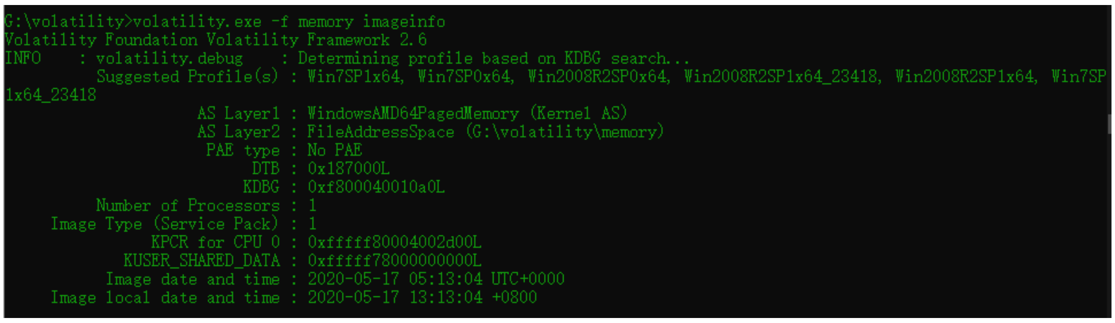
FitQR® PLus Decoder Report		
返回 | 首页		
Uploaded Status	Save.	Preview
Uploaded files count	1	
Uploaded file size	900	
Uploaded file name	pic_20200512170902.png	
Uploader elapsed	15622us (15.622ms)	
Decoded Status	Decoded	
Text Charset	GB18030	
HXDecoder elapsed	0us (0ms)	
Decoded Text	我就是弗拉格 Dozerctf{Congratulations_U_find_it}	
返回 | 首页		

flag：Dozerctf{Congratulations_U_find_it}

3. easy_analysis（难度：难）

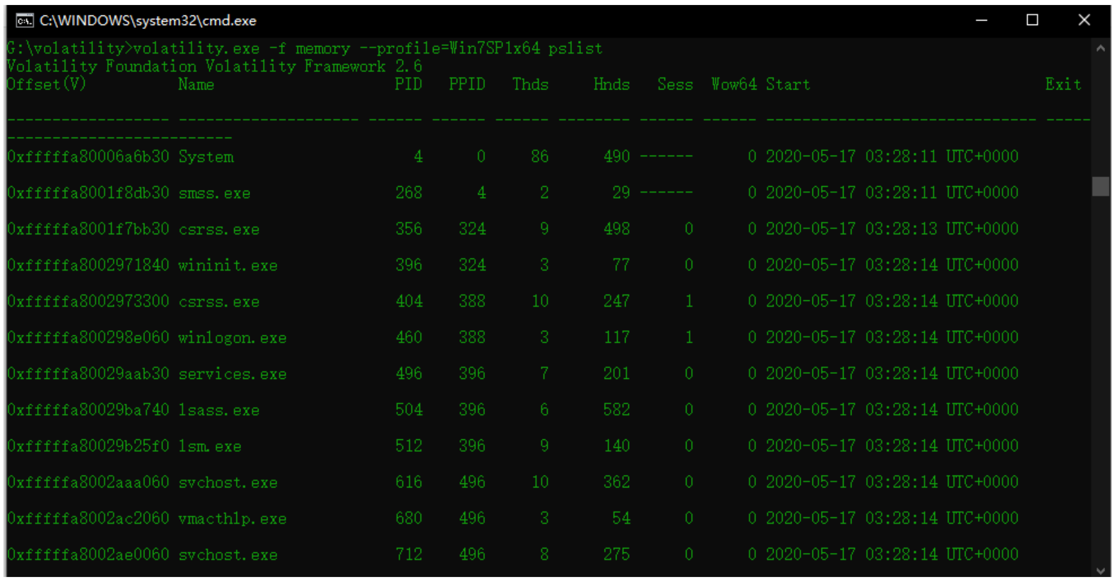
注：此处 volatility 使用 windows 版，linux 自行修改命令

使用 volatility.exe -f memory imageinfo 判断系统，猜测为 win7SP1X64



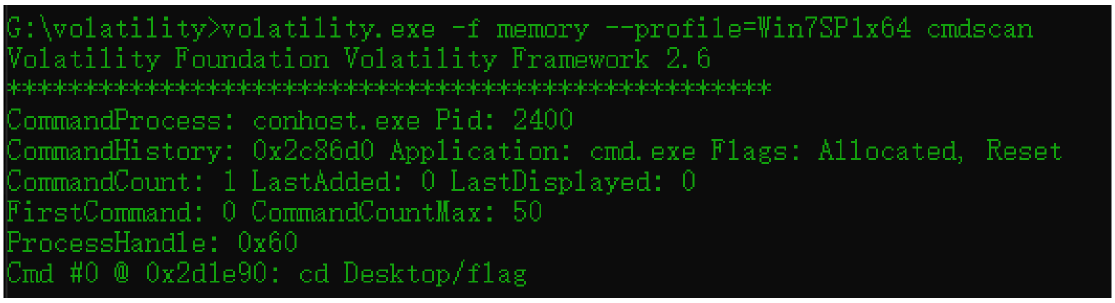
```
G:\volatility>volatility.exe -f memory imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP
1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (G:\volatility\memory)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf800040010a0L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff80004002d00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2020-05-17 05:13:04 UTC+0000
     Image local date and time : 2020-05-17 13:13:04 +0800
```

volatility.exe -f memory --profile=Win7SP1x64 pslist 查看进程



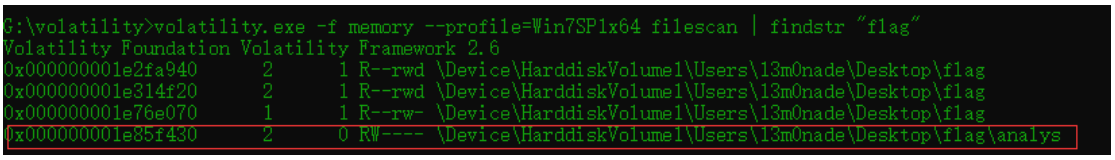
```
C:\WINDOWS\system32\cmd.exe
G:\volatility>volatility.exe -f memory --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit

------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ -----
-------------------------
0xfffffa80006a6b30 System                    4      0     86      490 ------      0 2020-05-17 03:28:11 UTC+0000

0xfffffa8001f8db30 smss.exe                268      4      2       29 ------      0 2020-05-17 03:28:11 UTC+0000

0xfffffa8001f7bb30 csrss.exe               356    324      9      498      0      0 2020-05-17 03:28:13 UTC+0000

0xfffffa8002971840 wininit.exe             396    324      3       77      0      0 2020-05-17 03:28:14 UTC+0000

0xfffffa8002973300 csrss.exe               404    388     10      247      1      0 2020-05-17 03:28:14 UTC+0000

0xfffffa800298e060 winlogon.exe            460    388      3      117      1      0 2020-05-17 03:28:14 UTC+0000

0xfffffa80029aab30 services.exe            496    396      7      201      0      0 2020-05-17 03:28:14 UTC+0000

0xfffffa80029ba740 lsass.exe               504    396      6      582      0      0 2020-05-17 03:28:14 UTC+0000

0xfffffa80029b25f0 lsm.exe                 512    396      9      140      0      0 2020-05-17 03:28:14 UTC+0000

0xfffffa8002aaa060 svchost.exe             616    496     10      362      0      0 2020-05-17 03:28:14 UTC+0000

0xfffffa8002ac2060 vmacthlp.exe            680    496      3       54      0      0 2020-05-17 03:28:14 UTC+0000

0xfffffa8002ae0060 svchost.exe             712    496      8      275      0      0 2020-05-17 03:28:14 UTC+0000
```

使用 volatility.exe -f memory --profile=Win7SP1x64 cmdscan 查看命令行记录，发现 flag 文件夹。



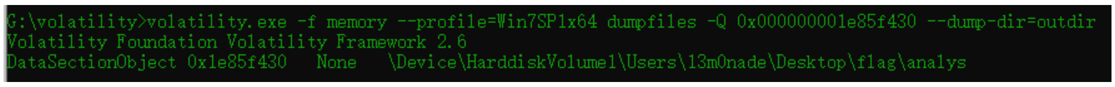
```
G:\volatility>volatility.exe -f memory --profile=Win7SP1x64 cmdscan
Volatility Foundation Volatility Framework 2.6
**************************************************
CommandProcess: conhost.exe Pid: 2400
CommandHistory: 0x2c86d0 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x60
Cmd #0 @ 0x2d1e90: cd Desktop/flag
```

G:\volatility>volatility.exe -f memory --profile=Win7SP1x64 filescan|findstr "flag"尝试查找带 flag 的文件发现一个 analyse.zip 文件

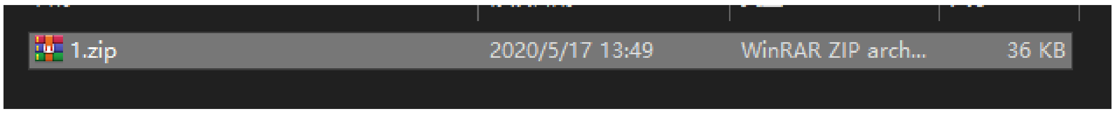


```
G:\volatility>volatility.exe -f memory --profile=Win7SP1x64 filescan | findstr "flag"
Volatility Foundation Volatility Framework 2.6
0x000000001e2fa940      2      1 R--rwd \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag
0x000000001e314f20      2      1 R--rwd \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag
0x000000001e76e070      1      1 R--rw- \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag
0x000000001e85f430      2      0 RW---- \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag\analys
```

volatility.exe -f memory --profile=Win7SP1x64 dumpfiles -Q 0x000000001e85f430 --dump-dir=outdir 导出文件，修改文件名

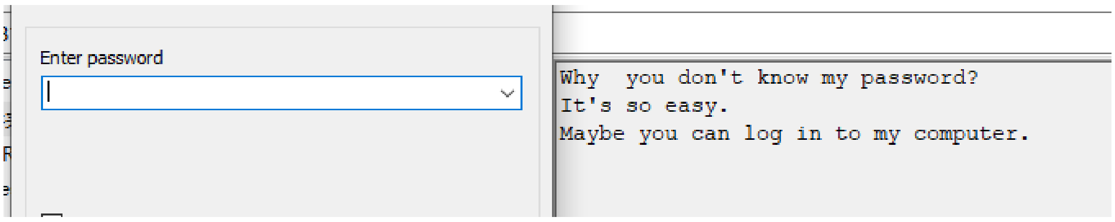


```
G:\volatility>volatility.exe -f memory --profile=Win7SP1x64 dumpfiles -Q 0x000000001e85f430 --dump-dir=outdir
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x1e85f430   None   \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag\analys
```



1.zip	2020/5/17 13:49	WinRAR ZIP arch...	36 KB

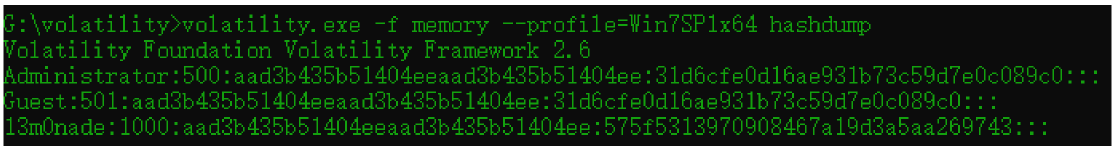
根据提示查找密码，猜测密码为用户登陆密码



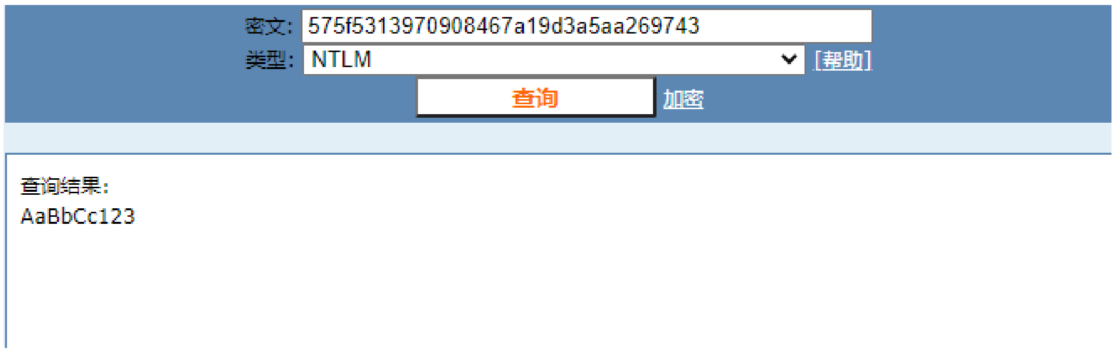
Enter password

```
Why  you don't know my password?
It's so easy.
Maybe you can log in to my computer.
```

使用 volatility.exe -f memory --profile=Win7SP1x64 hashdump 查看，解出 NTLM



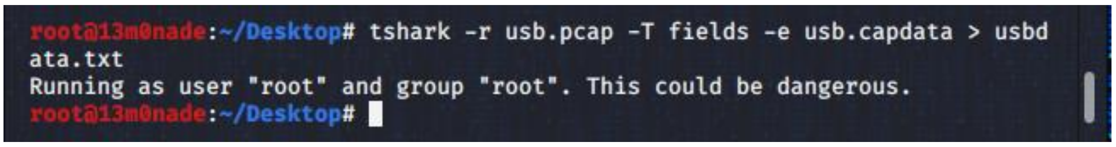
```
G:\volatility>volatility.exe -f memory --profile=Win7SP1x64 hashdump
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
13m0nade:1000:aad3b435b51404eeaad3b435b51404ee:575f5313970908467a19d3a5aa269743:::
```



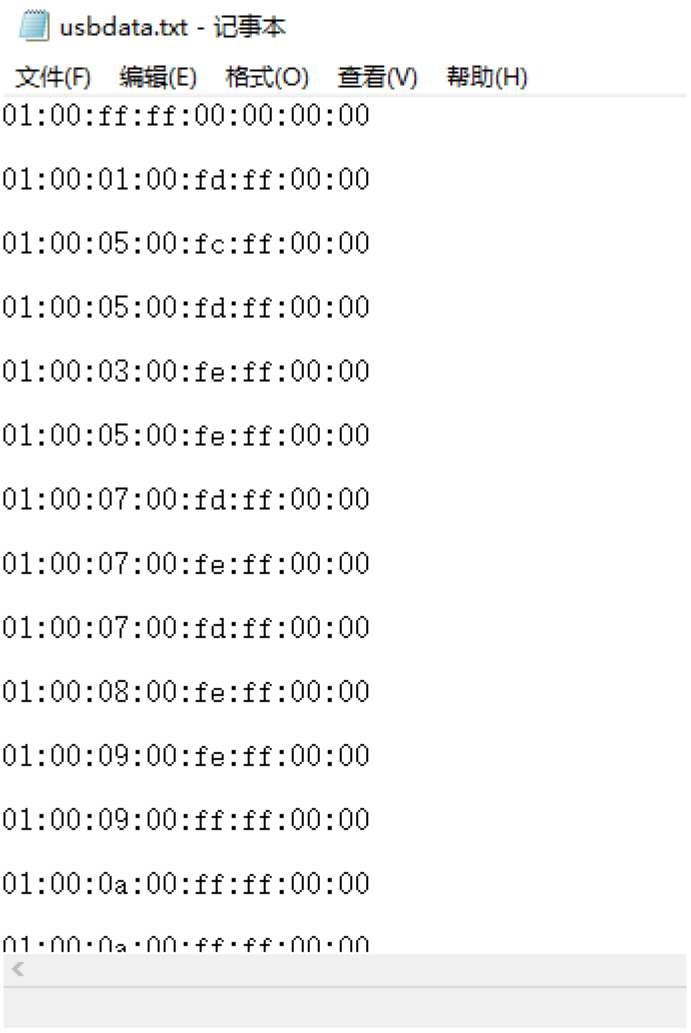
密文: 575f5313970908467a19d3a5aa269743
类型: NTLM

查询结果:
AaBbCc123

解压得到一个 usb 流量包，分析得



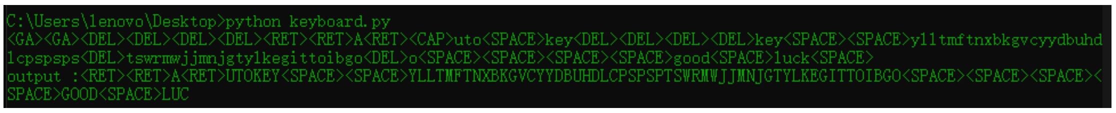
```
root@13m0nade:~/Desktop# tshark -r usb.pcap -T fields -e usb.capdata > usbd
ata.txt
Running as user "root" and group "root". This could be dangerous.
root@13m0nade:~/Desktop# 
```



```
01:00:ff:ff:00:00:00:00

01:00:01:00:fd:ff:00:00

01:00:05:00:fc:ff:00:00

01:00:05:00:fd:ff:00:00

01:00:03:00:fe:ff:00:00

01:00:05:00:fe:ff:00:00

01:00:07:00:fd:ff:00:00

01:00:07:00:fe:ff:00:00

01:00:07:00:fd:ff:00:00

01:00:08:00:fe:ff:00:00

01:00:09:00:fe:ff:00:00

01:00:09:00:ff:ff:00:00

01:00:0a:00:ff:ff:00:00

01:00:0a:00:ff:ff:00:00
```

运行脚本得到键盘记录

AUTOKEY   YLLTMFTNXBKGVCYYDBUHDLCPSPSPTSWRMWJJMNJGTYLKEGITTOIBGO

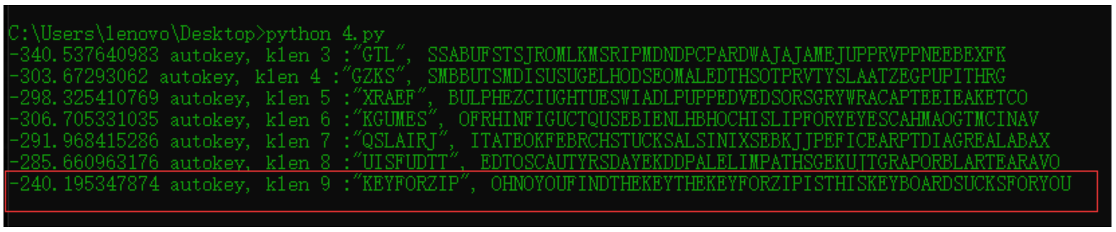


```
C:\Users\lenovo\Desktop>python keyboard.py
<GA><GA><DEL><DEL><DEL><DEL><RET><RET>A<RET><CAP>uto<SPACE>key<DEL><DEL><DEL><DEL>key<SPACE><SPACE>ylltmftnxbkgvcyydbuhd
lcpspsps<DEL>tswrmwjjmnjgtylkegittoibgo<DEL>o<SPACE><SPACE><SPACE><SPACE>good<SPACE>luck<SPACE>
output :<RET><RET>A<RET>UTOKEY<SPACE><SPACE>YLLTMFTNXBKGVCYYDBUHDLCPSPSPTSWRMWJJMNJGTYLKEGITTOIBGO<SPACE><SPACE><SPACE><
SPACE>GOOD<SPACE>LUC
```

对于自动密钥进行暴破

代码详见

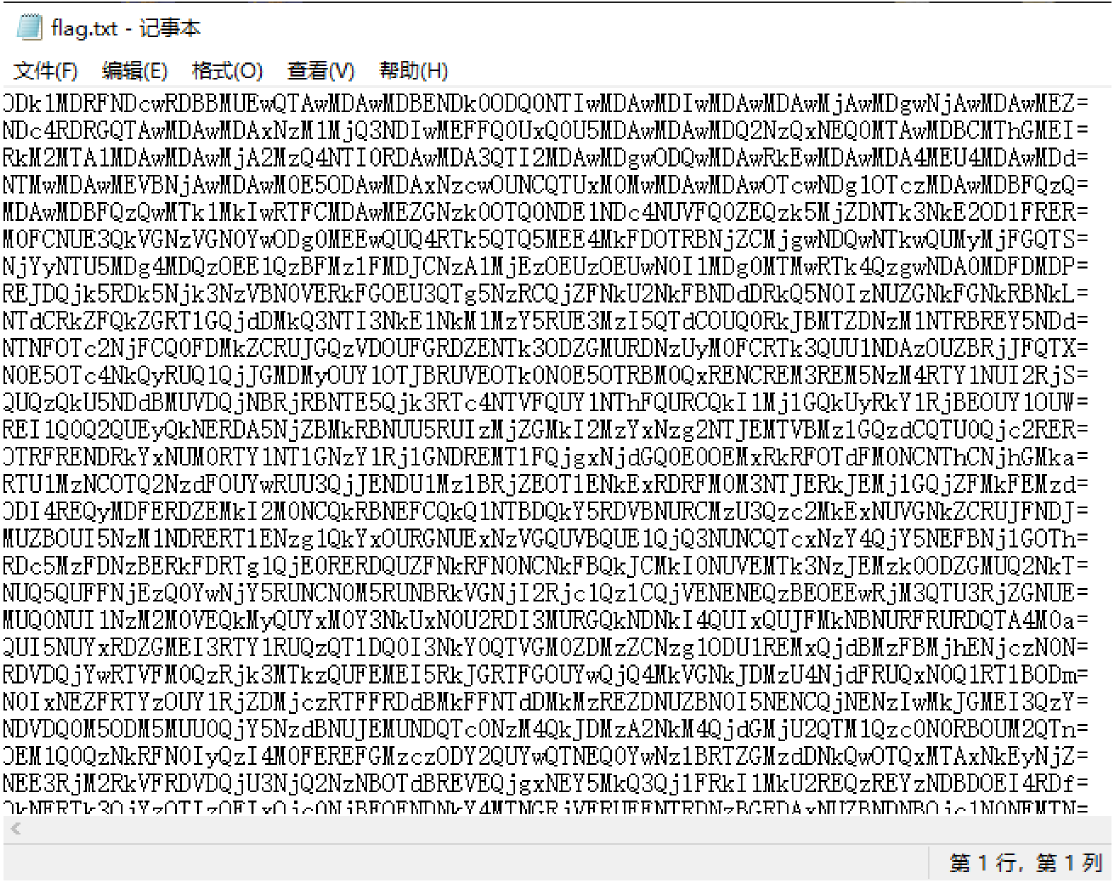
http://www.practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-autokey-cipher/



```
C:\Users\lenovo\Desktop>python 4.py
-340.537640983 autokey, klen 3 :"GTL", SSABUFSTSJROMLKMSRIPMDNDPCPARDWAJAJAMEJUPPRVPPNEEBEXFK
-303.67293062 autokey, klen 4 :"GZKS", SMBBUTSMDISUSUGELHODSEOMALEDTHSOTPRVTYSLAATZEGPUPITHRG
-298.325410769 autokey, klen 5 :"XRAEF", BULPHEZCIUGHTUESWIADLPUPPEDVEDSORSGRYWRACAPTEEIEAKETCO
-306.705331035 autokey, klen 6 :"KGUMES", OFRHINFIGUCTQUSEBIENLHBHOCHISLIPFORYEYESCAHMAOGTMCINAV
-291.968415286 autokey, klen 7 :"QSLAIRJ", ITATEOKFEBRCHSTUCKSALSINIXSEBKJJPEFICEARPTDIAGREALABAX
-285.660963176 autokey, klen 8 :"UISFUDTT", EDTOSCAUTYRSDAYEKDDPALELIMPATHSGEKUJTGRAPORBLARTEARAVO
-240.195347874 autokey, klen 9 :"KEYFORZIP", OHNOYOUFINDTHEKEYTHEKEYFORZIPISTHISKEYBOARDSUCKSFORYOU
```

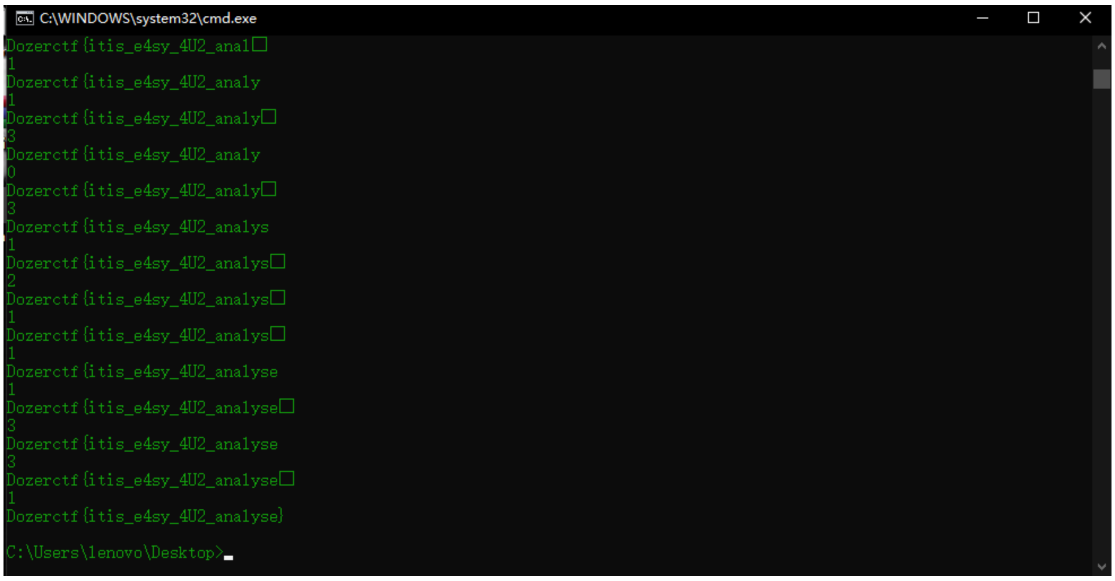
压缩包密码：thiskeyboardsucksforyou

得到的 flag.txt 是 base64 隐写



```
Dk1MDRFNDcwRDBBMUEwQTAwMDAwMDBENDk0ODQ0NTIwMDAwMDIwMDAwMDAwMjAwMDgwNjAwMDAwMEZ=
NDc4RDRGQTAwMDAwMDAxNzM1MjQ3NDIwMEFFQ0UxQ0U5MDAwMDAwMDQ2NzQxNEQ0MTAwMDBCMThGMEI=
RkM2MTA1MDAwMDAwMjA2MzQ4NTI0RDAwMDA3QTI2MDAwMDgwODQwMDAwRkEwMDAwMDA4MEU4MDAwMDd=
NTMwMDAwMEVBNjAwMDAwM0E5ODAwMDAxNzcwOUNCQTUxM0MwMDAwMDAwOTcwNDg1OTczMDAwMDBFQzQ=
MDAwMDBFQzQwMTk1MkIwRTFCMDAwMEZGNzk0OTQ0NDE1NDc4NUVFQ0ZEQzk5MjZDNTk3NkE2OD1FRER=
M0FCNUE3QkVGNzVGN0YwODg0MEEwQUQ4RTk5QTQ5MEE4MkFDOTRBNjZCMjgwNDQwNTkwQUMyMjFGQTS=
NjYyNTU5MDg4MDQzOEE1QzBFMz1FMDJCNzA1MjEzOEUzOEUwN0I1MDg0MTMwRTk4QzgwNDA0MDFDMDP=
REJDQjk5RDk5Njk3NzVBN0VERkFGOEU3QTg5NzRCQjZFNkU2NkFBNDdDRkQ5N0IzNUZGNkFGNkRBNkL=
NTdCRkZFQkZGRT1GQjdDMkQ3NTI3NkE1NkM1MzY5RUE3MzI5QTdCOUQ0RkJBMTZDNzM1NTRBREY5NDd=
NTNFOTc2NjFCQ0FDMkZCRUJGQzVDOUFGRDZENTk3ODZGMURDNzUyM0FCRTk3QUU1NDAzOUZBRjJFQTX=
N0E5OTc4NkQyRUQ1QjJGMDMyOUY1OTJBRUVEOTk0N0E5OTRBM0QxRENCREM3REM5NzM4RTY1NUI2RjS=
QUQzQkU5NDdBMUVDQjNBRjRBNTE5Qjk3RTc4NTVFQUY1NThFQURCQkI1Mj1GQkUyRkY1RjBEOUY1OUW=
REI1Q0Q2QUEyQkNERDA5NjZBMkRBNUU5RUIzMjZGMkI2MzYxNzg2NTJEMTVBMz1GQzdCQTU0Qjc2RER=
TRFRENDRkYxNUM0RTY1NT1GNzY1Rj1GNDREMT1FQjgxNjdGQ0E0OEMxRkRFOTdFM0NCNThCNjhGMka=
RTU1MzNCOTQ2NzdFOUYwRUU3QjJENDU1Mz1BRjZEOT1ENkExRDRFM0M3NTJERkJEMj1GQjZFMkFEMzd=
DI4REQyMDFERDZEMkI2M0NCQkRBNEFCQkQ1NTBDQkY5RDVBNURCMzU3Qzc2MkExNUVGNkZCRUJFNDJ=
MUZBOUI5NzM1NDRERT1ENzg1QkYxOURGNUExNzVGQUVBQUE1QjQ3NUNCQTcxNzY4QjY5NEFBNj1GOTh=
RDc5MzFDNzBERkFDRTg1QjE0RERDQUZFNkRFN0NCNkFBQkJCMkI0NUVEMTk3NzJEMzk0ODZGMUQ2NkT=
NUQ5QUFFNjEzQ0YwNjY5RUNCN0M5RUNBRkVGNjI2Rjc1Qz1CQjVENENEQzBEOEEwRjM3QTU3RjZGNUE=
MUQ0NUI1NzM2M0VEQkMyQUYxM0Y3NkUxN0U2RDI3MURGQkNDNkI4QUIxQUJFMkNBNURFRURQTA4M0a=
QUI5NUYxRDZGMEI3RTY1RUQzQT1DQ0I3NkY0QTVGM0ZDMzZCNzg1ODU1REMxQjdBMzFBMjhENjczN0N=
RDVDQjYwRTVFM0QzRjk3MTkzQUFEMEI5RkJGRTFGOUYwQjQ4MkVGNkJDMzU4NjdFRUQxN0Q1RT1BODm=
N0IxNEZFRTYzOUY1RjZDMjczRTFFRDdBMkFFNTdDMkMzREZDNUZBN0I5NENCQjNENzIwMkJGMEI3QzY=
NDVDQ0M5ODM5MUU0QjY5NzdBNUJEMUNDQTc0NzM4QkJDMzA2NkM4QjdGMjU2QTM1Qzc0N0RBOUM2QTn=
EM1Q0QzNkRFN0IyQzI4M0FEREFGMzczODY2QUYwQTNEQ0YwNz1BRTZGMzdDNkQwOTQxMTAxNkEyNjZ=
NEE3RjM2RkVFRDVDQjU3NjQ2NzNBOTdBREVEQjgxNEY5MkQ3Qj1FRkI1MkU2REQzREYzNDBDOEI4RDf=
kNERTk3QjYzOTIzOEIxQjc0NjBEOENDNkY4MTNGRjVERUEENTRDNzBGRDAxNUZBNDNBQjc1NONEMTN=
```

运行脚本得到 flag


C:\WINDOWS\system32\cmd.exe

```
Dozerctf{itis_e4sy_4U2_anal□
1
Dozerctf{itis_e4sy_4U2_analy
1
Dozerctf{itis_e4sy_4U2_analy□
3
Dozerctf{itis_e4sy_4U2_analy
0
Dozerctf{itis_e4sy_4U2_analy□
3
Dozerctf{itis_e4sy_4U2_analys
1
Dozerctf{itis_e4sy_4U2_analys□
2
Dozerctf{itis_e4sy_4U2_analys□
1
Dozerctf{itis_e4sy_4U2_analys□
1
Dozerctf{itis_e4sy_4U2_analyse
1
Dozerctf{itis_e4sy_4U2_analyse□
3
Dozerctf{itis_e4sy_4U2_analyse
3
Dozerctf{itis_e4sy_4U2_analyse□
1
Dozerctf{itis_e4sy_4U2_analyse}

C:\Users\lenovo\Desktop>
```

代码

```
def get_base64_diff_value(s1, s2):
    base64chars                                                        =
’ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/’
    res = 0
    for i in xrange(len(s2)):
        if s1[i] != s2[i]:
            return abs(base64chars.index(s1[i]) - base64chars.index(s2[i]))
    return res


def solve_stego():
    with open(’flag.txt’, ’rb’) as f:
        file_lines = f.readlines()
        bin_str = ’’
        for line in file_lines:
            steg_line = line.replace(’\n’, ’’)
            norm_line                    =                   line.replace(’\n’,
’’).decode(’base64’).encode(’base64’).replace(’\n’, ’’)
            diff = get_base64_diff_value(steg_line, norm_line)
            print diff
            pads_num = steg_line.count('=')
            if diff:
                bin_str += bin(diff)[2:].zfill(pads_num * 2)
            else:
                bin_str += '0' * pads_num * 2
            print goflag(bin_str)


def goflag(bin_str):
    res_str = ''
    for i in xrange(0, len(bin_str), 8):
        res_str += chr(int(bin_str[i:i + 8], 2))
    return res_str


if __name__ == '__main__':
    solve_stego()
```

flag：Dozerctf{itis_e4sy_4U2_analyse}