强网杯2021——wp

原创

Em0s_Er1t 于 2021-11-30 18:03:29 发布 2948 收藏

分类专栏： CTF-MISC 文章标签： 安全 安全漏洞 反汇编

本文链接：https://blog.csdn.net/m0_46296905/article/details/121639142

文章目录

一、MISC

（一）ISO1995

> 题目描述：We follow ISO1995. ISO1995 has many problems though. One known problem is a time. 压缩包解压密码：fantasicqwb2021

光看文件头辨别不出什么文件，file命令看一下发现是Linux Debian的磁盘文件

binwalk提取出一个ISO文件，用ultraiso打开发现每个字符都分开到一个文件中，直接全部提取出来写个python脚本拼接

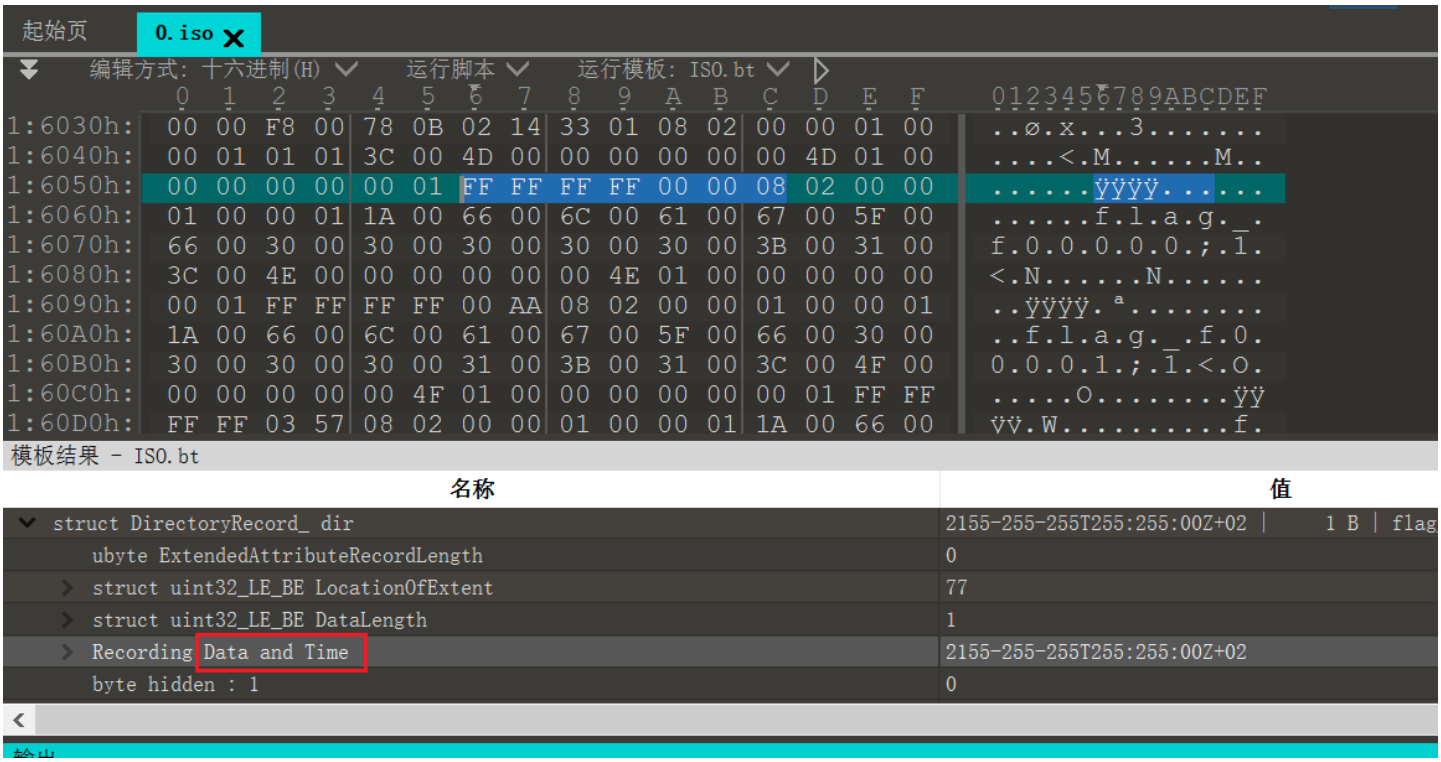
```
flag=''
for i in range(1024):
    name='flag_f'+str(i).rjust(5,'0')
    #print(name)
    with open("E:/CTF_project/Game/强网杯2021/{}".format(name),"r") as fr:
        flag+=fr.readline()
print(flag)
```

得到下面字符串

```
!=gF~B.@YB01.%DYzb^-1}jH&@,K[7t/LOi*5b)L'<pW'am\W4LH@\toGKE1{"oDW0qf2{1{W_0V-m:af8AO4^iCT_+ $W3cz(LO)L_-s8'_<Ic/
KFP9vrr~6\ni{~#g5cs#7z2s++Y1BbYQV'iSl=DZ__|3T1QxWEwX}NJ@_3SdKK]91b?s-rS6gQwBs@4#5IGxW#&ArDw~_x"!_I^O`x5o'.s5)+c9
RU'/%_b[rjiOP0y!&/)WjKR#IjWh0\,Dr!@PH^Nf%,YoWEJ(2wXj/u~Y@gh%&_Gz5U`A=0pAV$E/ >1Kg:@4tS:V4ZB`1_x*.17B&:<xn0rW|2TY
_DSN<zvbKCj7+6w'r}Lo8':fYC@FvJ02VbO)noQlMI3#AZ+]U3##P|W{V>z,G5[s r|Ra~-P'>oYJ{O{B"oh=VG*SB2KoNt1 7|,*r#d!=N'%O$1
Tron:7}6C(yVSIZmtw8Xza]6D,nn*q&KHNK,PW .b<h E$){Kw_)h,=m41LAv'f6l:I xN:4z0{>&F5(cRg|:M9RMX $,8/1vq-][?a/H}1"X;((
,MZ(=WJ4o</_8.D9Q8~S"aA:RNTxpsC8LKW+Pfgw<NTqmy_8G6Np%c-9tAG-em&]1IYtz\IJa1KD&z<k'w7vH Fr--py2uH=;3l*iuisp39+m;"1
:xPJB@*LB8;x*?G.``n^[Pib$KM>RFG#vDrwlk@QC0ebUkG,~fw+xH[W<{:eJmcbx,Yi6KcZ~}vH_R,t{F =}gTKX&;^_Fv1b,DezJ1N}6q)76a]
Us=\u8tY;t*#}zSGo`-h64=u2bGZ)I(&%K68&!nQke&+gX=L4TmMy$5nHC&+#<486HKF4f0d%1?I:1=M[p~DxBLtCKh\>4<Qf+cj?a3p0F`4*-%%
7*<~'^+KkQ<*z9oUgrgO$:NC.Di<.$`s+69Pn7:IgO`^\T%n |Q'G&9Tx-@!6W<VK_5tH/#i>$7SKKH[Dki-o{b{?j?4.Zw+aV!|Zi{2oTqk*#!O
0h$-6oCbPpaZbPfi
```

根据题目的提示`One known problem is a time`，说明需要把这些文件按时间顺序排一下就能得到flag。那就定位到iso文件的目录记录字段，把这些日期时间字段都dump下来即可

发现前四个字节年、月、日、时设置的都是FF，最后一个GMT都设置的是0x08，那就取其中间的2个字段。



写exp

```
import re
time=[]
flag=''
with open(r"0.iso","rb") as fr:
    fr.seek(0x16043)
    s=fr.read()
    for i in range(len(s)):
        try:
            if s[i:i+4]==b'\xff\xff\xff\xff':
                time.append(int.from_bytes(s[i+4:i+6], byteorder='big', signed=False))
                name='flag_f'+str(time[-1]).rjust(5,'0')
                #print(name)
                with open("iso1995\{}".format(name),"r") as fr:
                    flag+=fr.readline()
        except:
            break
print(flag)
```

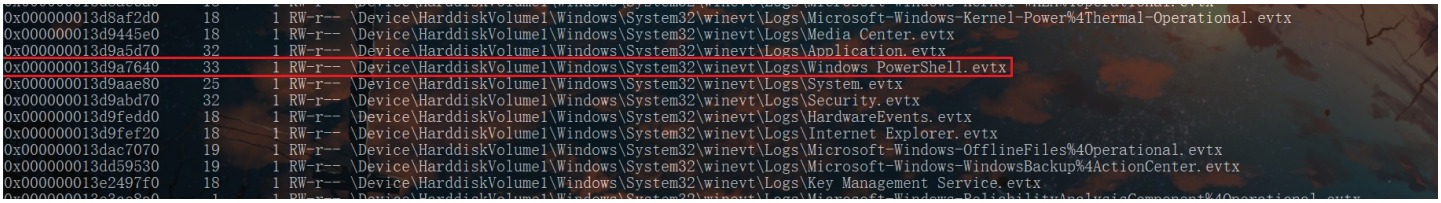
得到flag

```
FLAG{Dir3ct0ry_jYa_n41}
```

（二）BlueTeaming

> 题目描述：Powershell scripts were executed by malicious programs. What is the registry key that contained the power shellscript content?
> （恶意程序执行了一个Powershell脚本。包含power shell脚本内容的注册表项是什么？）

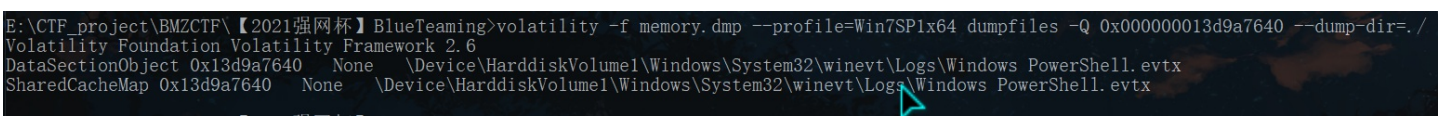
是一道内存取证题，既然是powershell脚本运行，那就找powershell日志（windows中的powershell日志文件一般为`WindowsPowerShell.evtx`），filescan确认一下是否存在



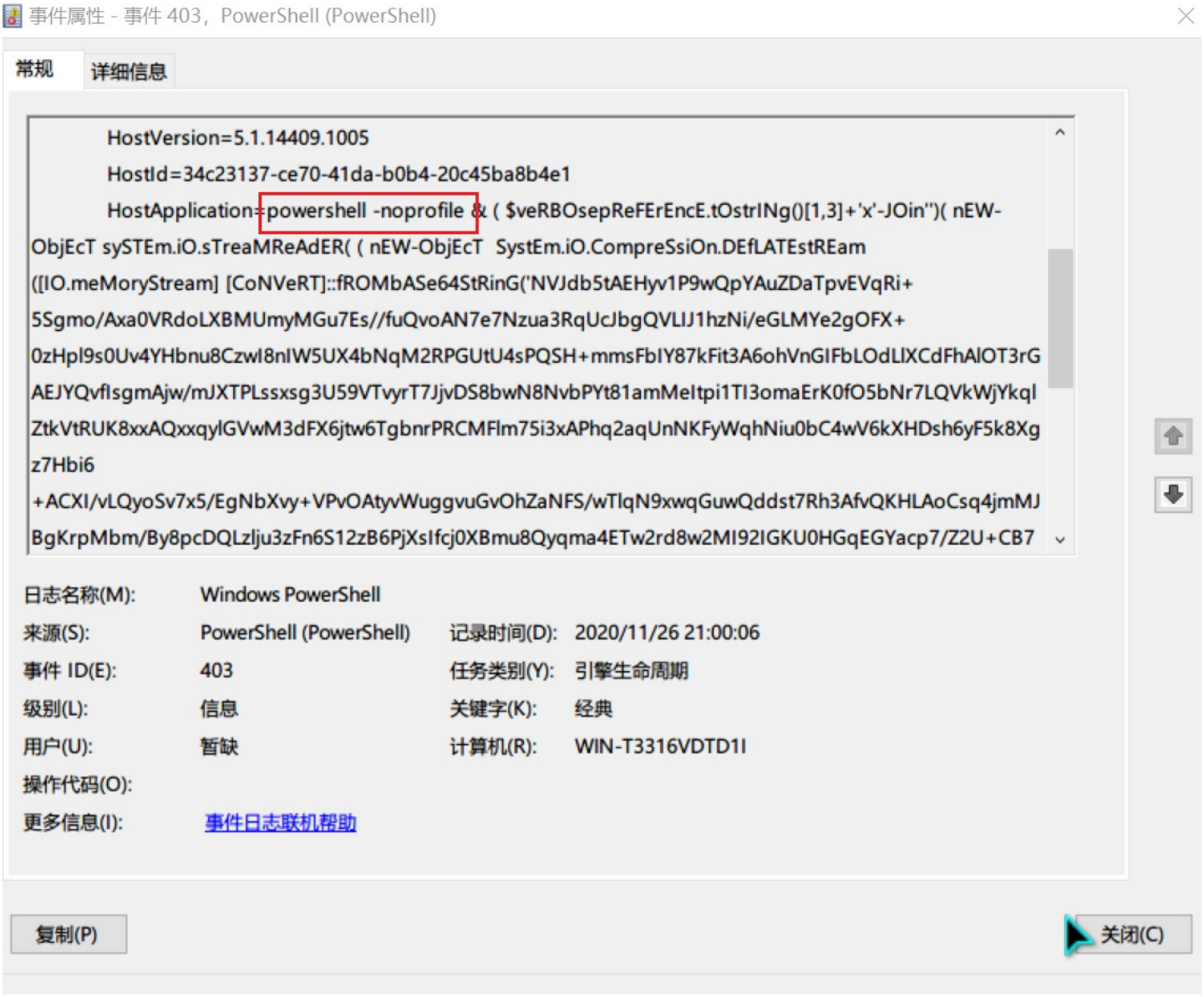

```
volatility -f memory.dmp --profile=Win7SP1x64 filescan | find "evtx"
```

然后把它dump出来

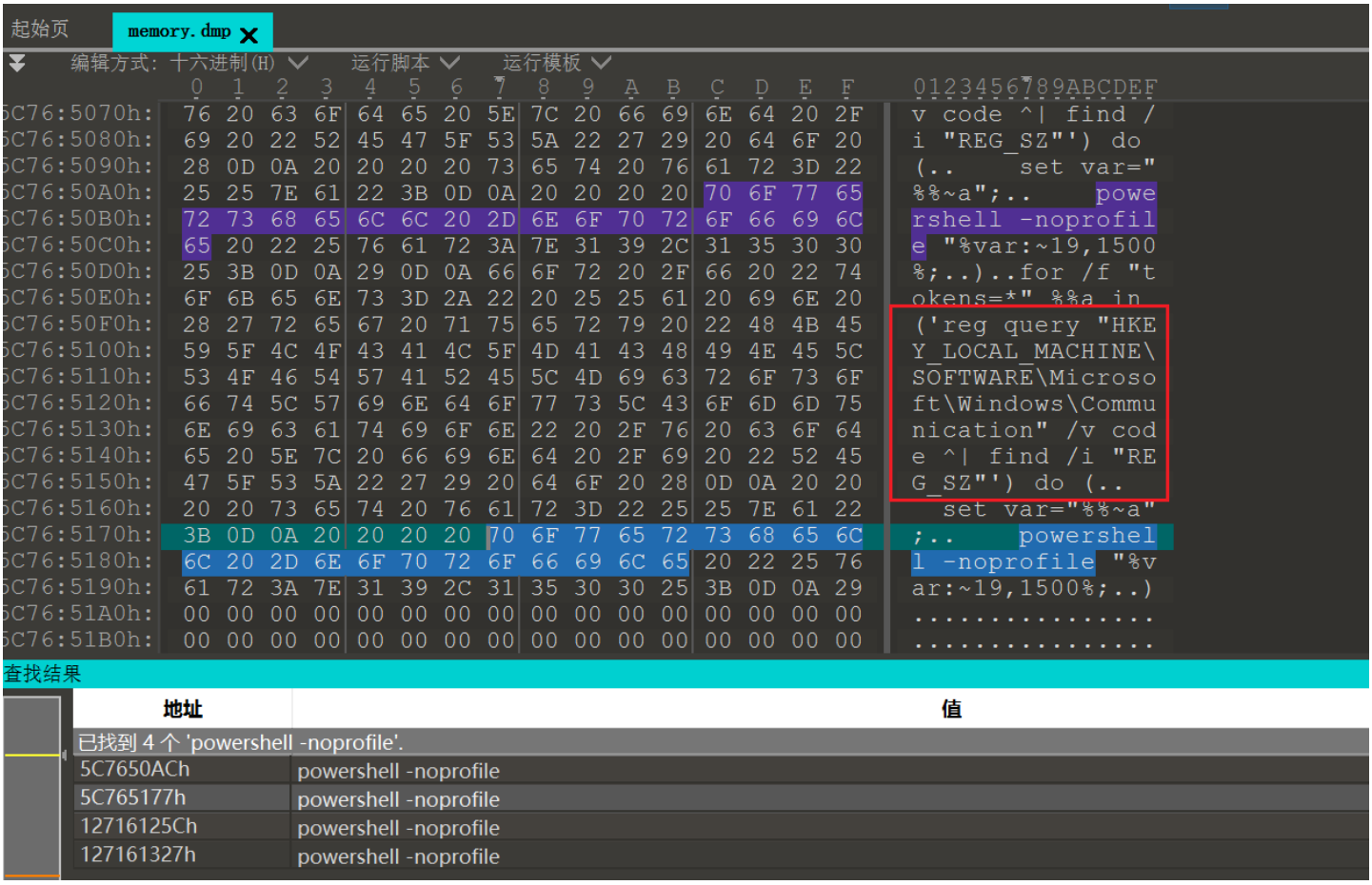
```
volatility -f memory.dmp --profile=Win7SP1x64 dumpfiles -Q 0x000000013d9a7640 --dump-dir=./
```



后缀名改`evtx`后在Windows事件查看器里分析，随便选一个事件进去可以看到在powershell中执行的命令关键字`powershell -noprofile`，后面跟的应该就是恶意脚本



010 Editor或者其它编辑器打开memory.dmp搜索powershell -noprofile即可看到注册表项。

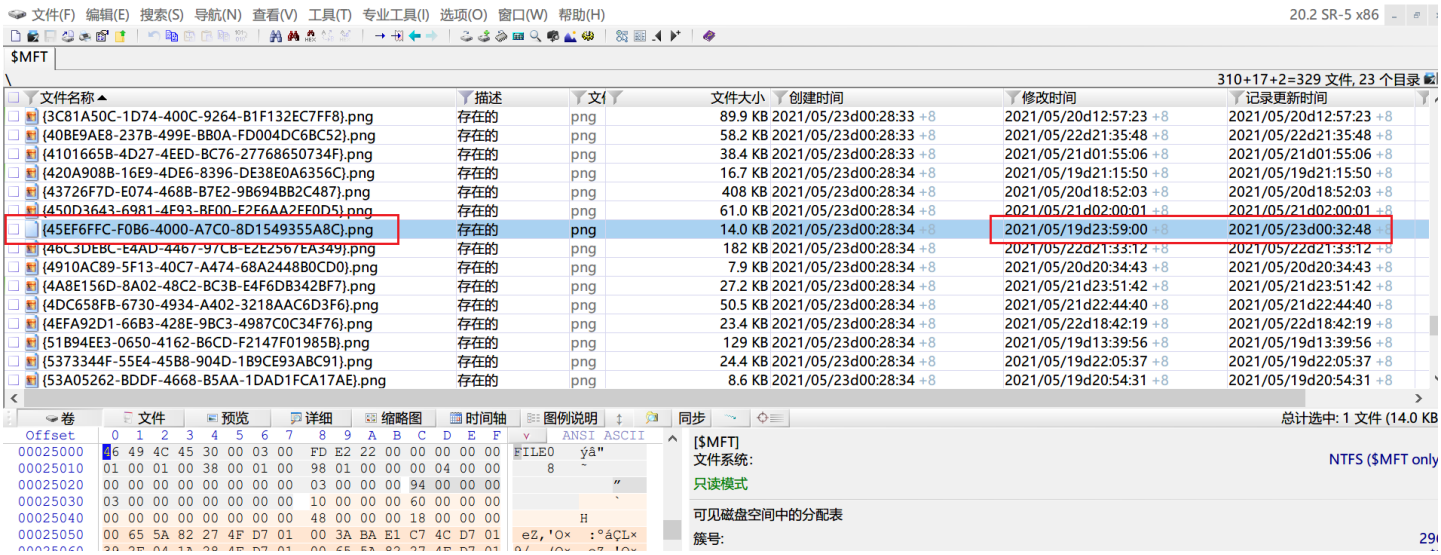



```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Communication
```

（三）EzTime

> 题目描述：Forensic.Find a file that a time attribute has been modified by a program.
> （查找时间属性已被程序修改的文件。）

X-ways打开，找修改时间与记录更新时间不一样的



得到flag

```
{45EF6FFC-F0B6-4000-A7C0-8D1549355A8C}.png
```

（四）Cipherman

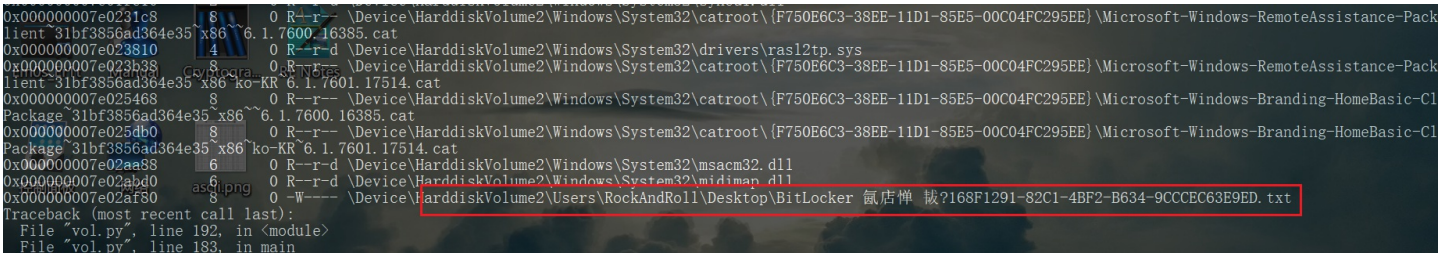
> 题目描述：The attacker maliciously accessed the user's PC and encrypted specific volumes. How to decrypto the volume?（攻击者恶意访问用户的PC并加密特定卷。如何解密该卷？）

内存取证

先扫一下文件

```
volatility -f memory --profile=Win7SP1x86_23418 filescan
```

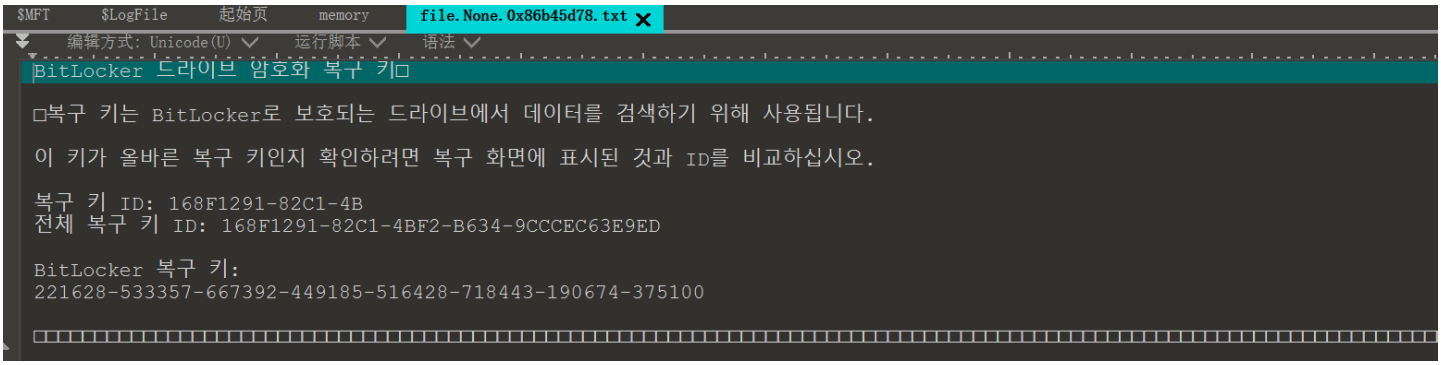
发现BitLocker恢复密钥

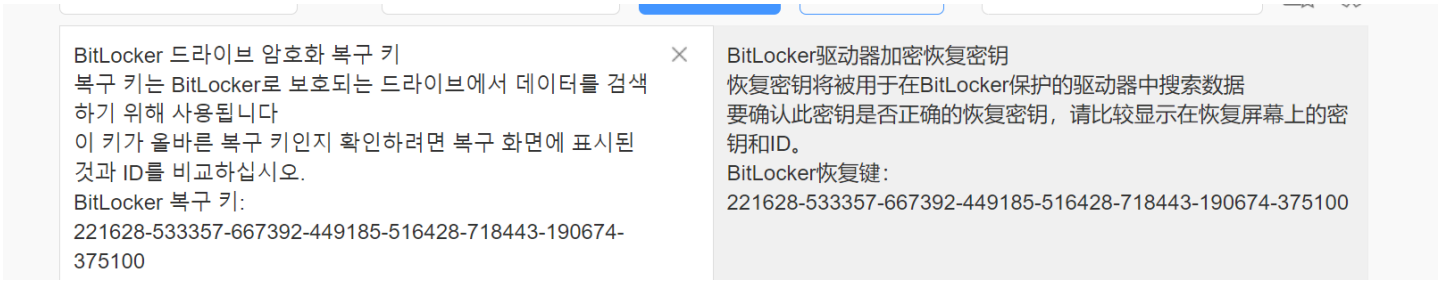


dump出来（不知道为啥windows下volatility报错，就换linux了）

```
vol.py -f memory --profile=Win7SP1x86_23418 dumpfiles -Q 0x000000007e02af80 --dump-dir=./
```

得到内容如下





恢复密钥就是

```
221628-533357-667392-449185-516428-718443-190674-375100
```

找个挂载工具挂载到本地，双击进入该盘会弹出要求输入PIN码，更多里面切换到输入恢复密钥然后输入即可

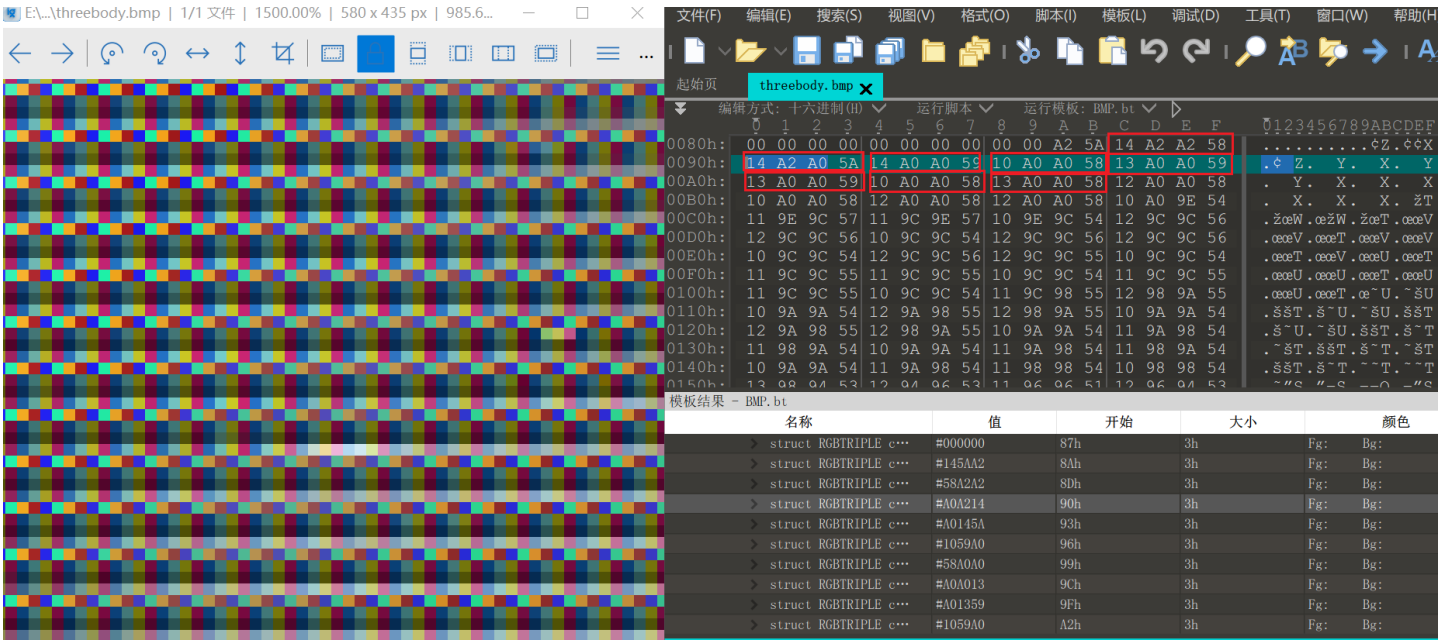
得到flag

```
Wow, you have a great ability. How did you solve this? Are you a hacker? Please give me a lesson later.
```

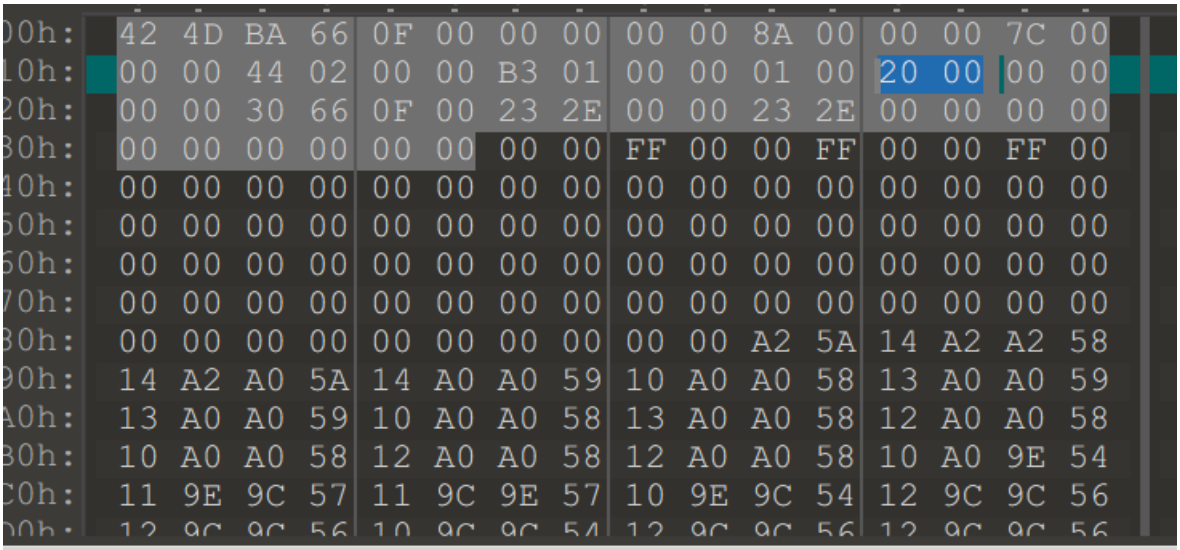
（五）threebody

> 参考：强网杯2021-threebody - wdxxg's Blog

了解完bmp图片结构之后再来看，发现这是一个未压缩且直接给出颜色数据的位图图片，深度是24，也就是3个字节代表一个像素，但010 Editor中观察到这些数据更像是4个字节一组（因为对于相邻的像素颜色值总是相近的）



所以修改biBitCount字段值为32，变成4个字节一个像素

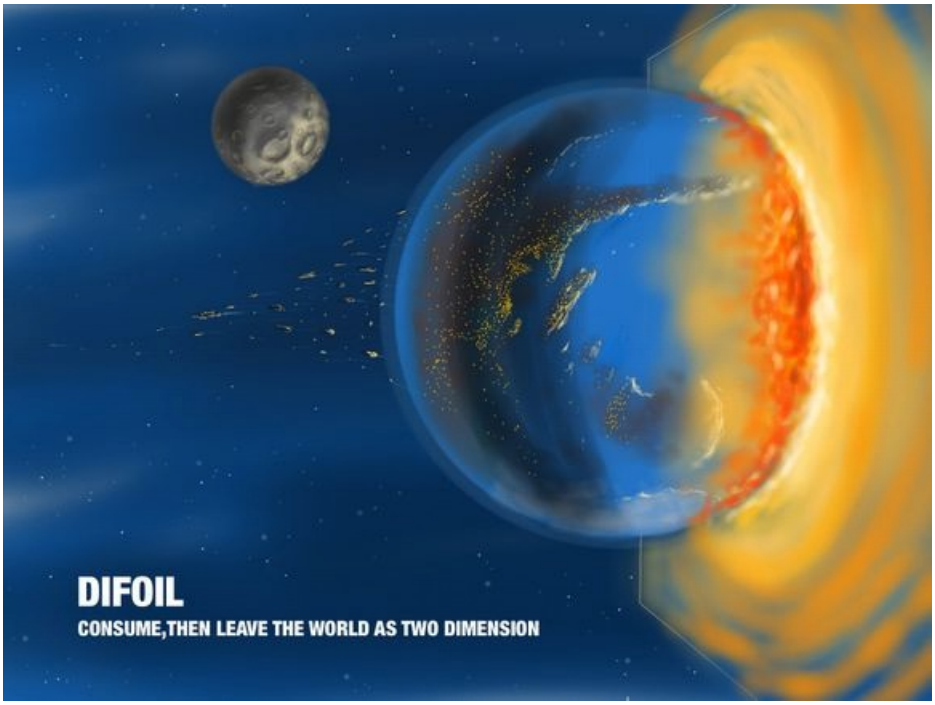


00h:	42	4D	BA	66	0F	00	00	00	00	00	8A	00	00	00	7C	00
10h:	00	00	44	02	00	00	B3	01	00	00	01	00	20	00	00	00
20h:	00	00	30	66	0F	00	23	2E	00	00	23	2E	00	00	00	00
30h:	00	00	00	00	00	00	00	00	FF	00	00	FF	00	00	FF	00
40h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80h:	00	00	00	00	00	00	00	00	00	00	A2	5A	14	A2	A2	58
90h:	14	A2	A0	5A	14	A0	A0	59	10	A0	A0	58	13	A0	A0	59
A0h:	13	A0	A0	59	10	A0	A0	58	13	A0	A0	58	12	A0	A0	58
B0h:	10	A0	A0	58	12	A0	A0	58	12	A0	A0	58	10	A0	9E	54
C0h:	11	9E	9C	57	11	9C	9E	57	10	9E	9C	54	12	9C	9C	56
D0h:	12	9C	9C	56	10	9C	9C	54	12	9C	9C	56	12	9C	9C	56

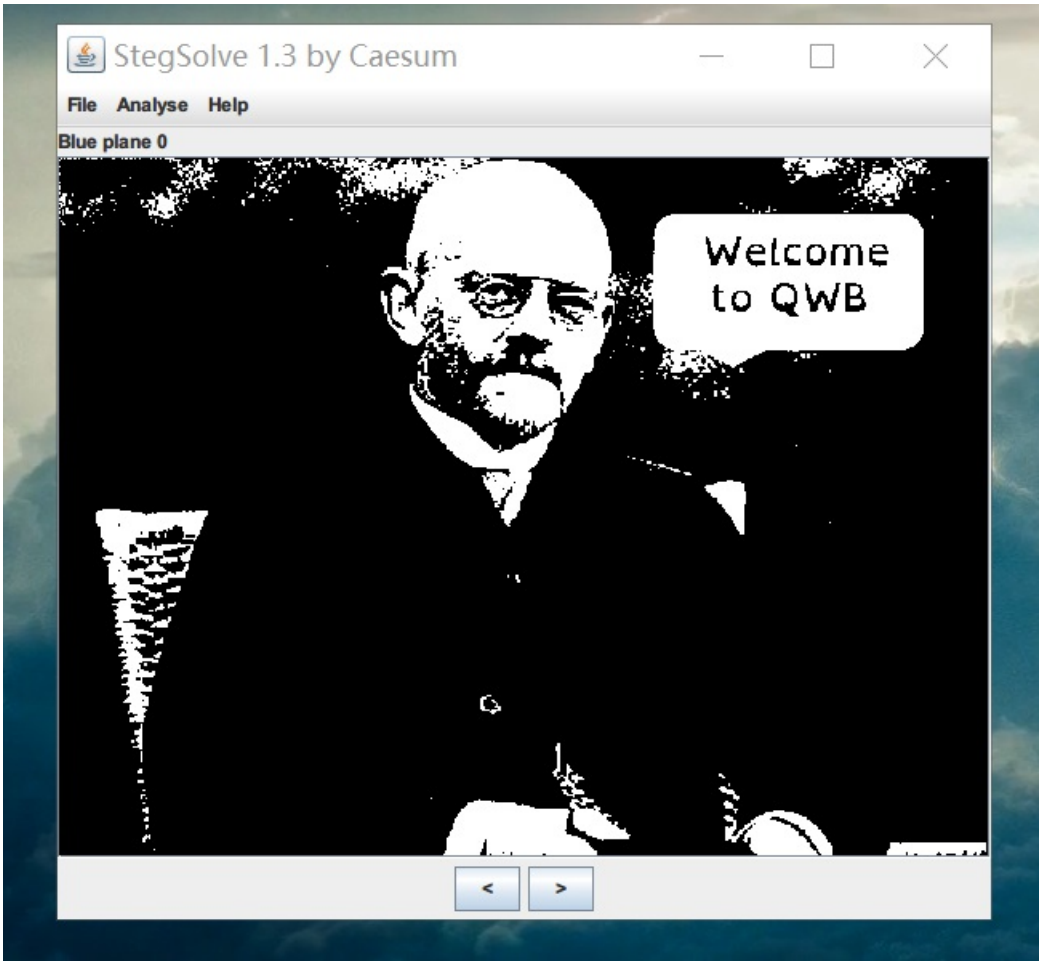
结果 - BMP.bt

名称	值	开始	大小
struct BITMAPINFOHEADER bmih		Eh	28h
DWORD biSize	124	Eh	4h
LONG biWidth	580	12h	4h
LONG biHeight	435	16h	4h
WORD biPlanes	1	1Ah	2h
WORD biBitCount	32	1Ch	2h
DWORD biCompression	0	1Eh	4h
DWORD biSizeImage	1009200	22h	4h
LONG biXPelsPerMeter	11811	26h	4h
LONG biYPelsPerMeter	11811	2Ah	4h

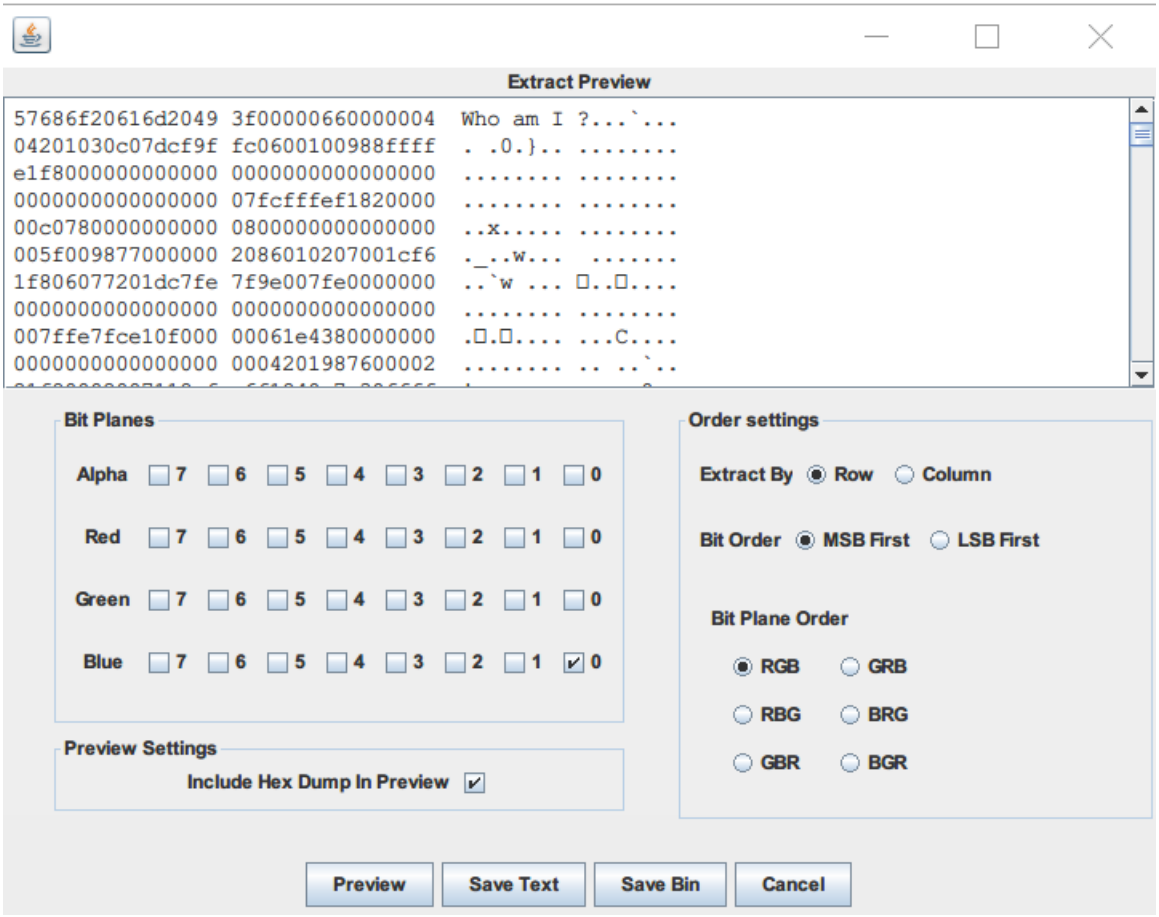
保存后得到的图片如下

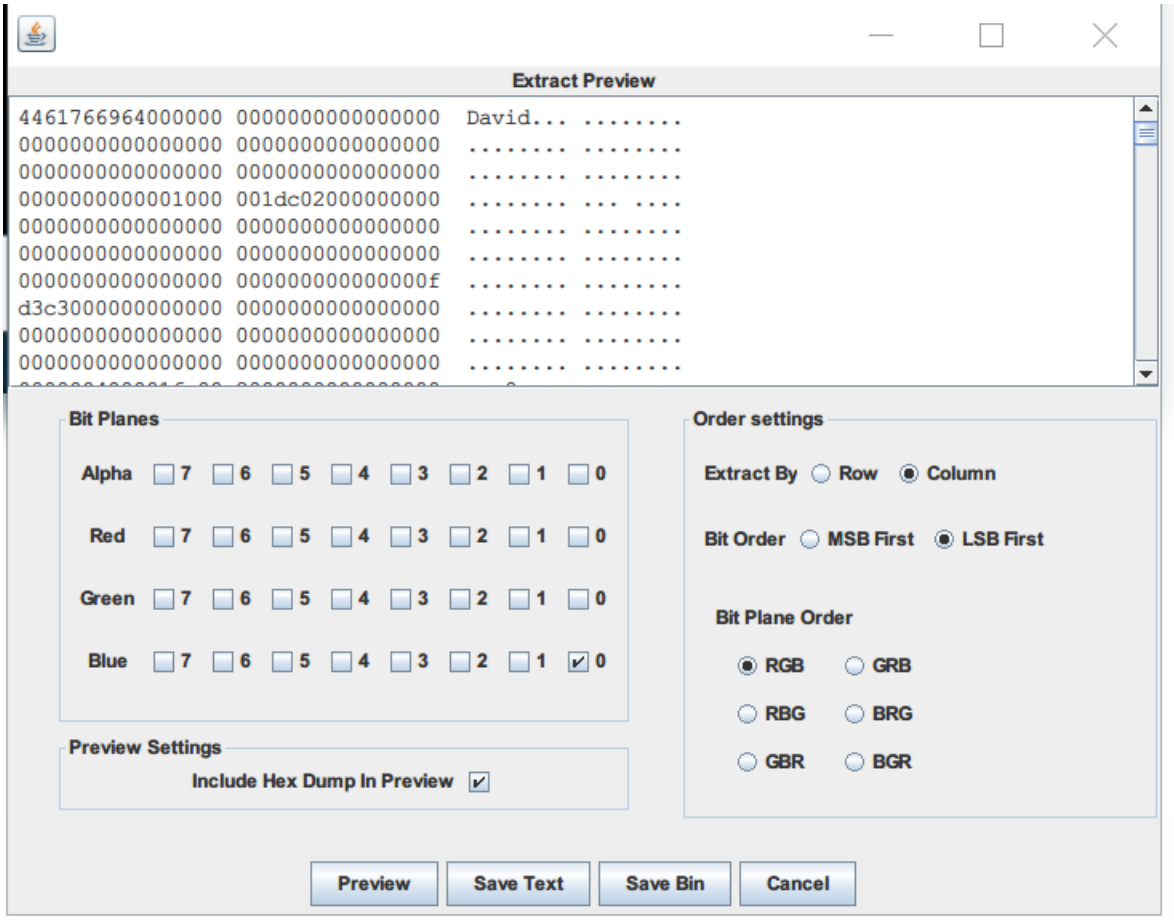


StegSolve中看RGB各个通道，只有这个正常些



行列分别有LSB隐写


Extract Preview
57686f20616d2049 3f00000660000004 Who am I ?...`...
04201030c07dcf9f fc0600100988ffff . .0.}.. ....
e1f8000000000000 0000000000000000 ....
0000000000000000 07fcfffef1820000 ....
00c0780000000000 0800000000000000 ..x..... ....
005f009877000000 2086010207001cf6 ._..w... ....
1f806077201dc7fe 7f9e007fe0000000 ..`w ... □..□....
0000000000000000 0000000000000000 ....
007ffe7fce10f000 00061e4380000000 .□.□.... ...C....
0000000000000000 0004201987600002 ....`..
Bit Planes
Alpha 7 6 5 4 3 2 1 0
Red 7 6 5 4 3 2 1 0
Green 7 6 5 4 3 2 1 0
Blue 7 6 5 4 3 2 1 0
Order settings
Extract By Row Column
Bit Order MSB First LSB First
Bit Plane Order
RGB GRB
RBG BRG
GBR BGR
Preview Settings
Include Hex Dump In Preview
Preview
Save Text
Save Bin
Cancel

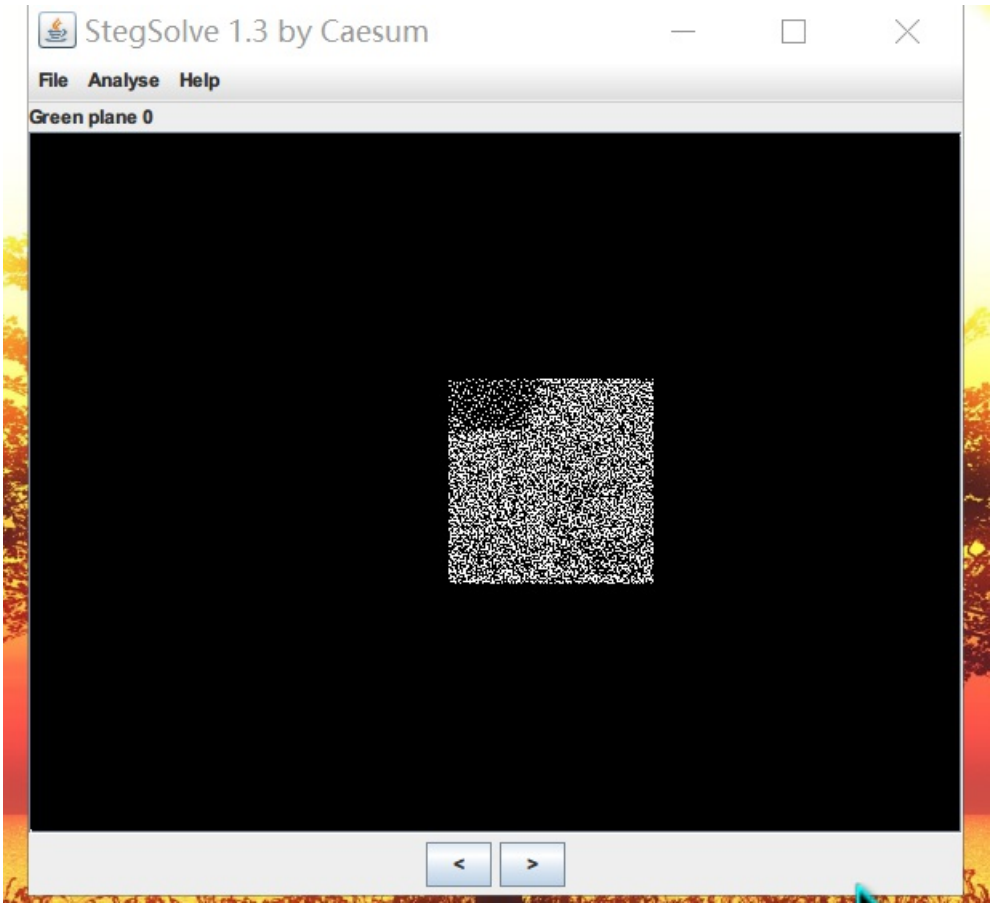

Extract Preview
4461766964000000 0000000000000000 David... ....
0000000000000000 0000000000000000 ....
0000000000000000 0000000000000000 ....
0000000000001000 001dc02000000000 ....
0000000000000000 0000000000000000 ....
0000000000000000 0000000000000000 ....
0000000000000000 000000000000000f ....
d3c3000000000000 0000000000000000 ....
0000000000000000 0000000000000000 ....
0000000000000000 0000000000000000 ....
Bit Planes
Alpha 7 6 5 4 3 2 1 0
Red 7 6 5 4 3 2 1 0
Green 7 6 5 4 3 2 1 0
Blue 7 6 5 4 3 2 1 0
Order settings
Extract By Row Column
Bit Order MSB First LSB First
Bit Plane Order
RGB GRB
RBG BRG
GBR BGR
Preview Settings
Include Hex Dump In Preview
Preview
Save Text
Save Bin
Cancel

网上搜一下David，查到这个人是戴维·希尔伯特，一个数学家

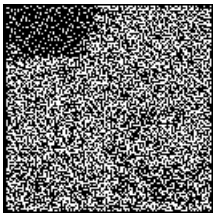
用StegSolver只能看到bmp的RGB三个通道，不能看到rgbReserved通道的情况（试过了，Alpha通道看到的是白屏），本来想找个在线工具把bmp转png，然后发现没有用，可能是那个网站转换算法有问题把，所以无奈之下写个脚本转换成png格式再去看各个通道的情况。

```
from PIL import Image
image=Image.new(mode='RGBA',size=(580,435))
with open(r'threebody.bmp','rb') as f:
    file=f.read()
    index=0
    for i in range(434,-1,-1):  #根据bmp的结构知道该bmp文件上下倒序存储像素值
        for j in range(0,580):
            s=[]
            for t in range(0,4):
                s.append(file[index])
                index+=1
            image.putpixel((j,i),(s[2],s[1],s[0],s[3]))  #
    image.show()
    image.save('threebody_new.png')
```

那个通道果然有问题，



我们先把上面的绿色通道得到的图片导出，稍微剪切一下



然后还有个线索没用到——希尔伯特，到搜索引擎里查希尔伯特与三体就找到这样一篇文章

真·降维打击！《三体》中二向箔吞噬地球的场景成真了！_腾讯新闻 (qq.com)

也就是说将上面的图片看作01矩阵，用希尔伯特曲线进行“降维打击”，变成一维的二进制流

贴上别的博主写的降维脚本学习一下

```
import numpy as np
from PIL import Image
#安装：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple hilbertcurve
from hilbertcurve.hilbertcurve import HilbertCurve
#提取像素数据
with Image.open('threebody_new.png') as img:
    arr = np.asarray(img)
arr = np.vectorize(lambda x: x&1)(arr[:,:,2])
#确定图片中的有效区域
for x1 in range(np.size(arr,0)):
    if sum(arr[x1])>0:
        break
for x2 in reversed(range(np.size(arr,0))):
    if sum(arr[x2])>0:
        break
for y1 in range(np.size(arr,1)):
    if sum(arr[:,y1])>0:
        break
for y2 in reversed(range(np.size(arr,1))):
    if sum(arr[:,y2])>0:
        break
#剪切出有效二维数据
arr = arr[x1:x2+1, y1:y2+1]
#print(x2+1-x1)#得出是128*128的矩阵
#构建希尔伯特曲线对象
hilbert_curve = HilbertCurve(7, 2)
#生成一维的二进制流数据
s = ''
for i in range(np.size(arr)):
    [x,y] = hilbert_curve.point_from_distance(i)
    s += str(arr[127-y][x])
#转ASCII文本写入文件
with open('output', 'wb') as f:
    f.write(int(s,2).to_bytes(2048, 'big'))
```

跑出来得到一个C源文件，稍微修改一下语法报错（malloc前面加(char *)）

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char* a(char* s);

char t[65536];

void b()
{
 printf("%s\n\nint main()\n{\n\tstrcpy(t, \"%s\");\n\tb();\n\treturn 0;\n}\n\nchar* a(char* s)\n{\n\tint l=strle
n(s);\n\tchar* n=malloc(l*2+1);\n\tchar* p=n;\n\tfor(int i=0; i<l; i++)\n\t\tswitch(s[i])\n\t\t{\n\t\t\tcase '\\
n':\n\t\t\t\t*p++='\\\\'; *p++='n'; break;\n\t\t\tcase '\\t':\n\t\t\t\t*p++='\\\\'; *p++='t'; break;\n\t\t\tcase
 '\\\\':\n\t\t\t\t*p++='\\\\'; *p++='\\\\'; break;\n\t\t\tcase '\\\"':\n\t\t\t\t*p++='\\\\'; *p++='\\\"'; break;
\n\t\t\tdefault:\n\t\t\t\t*p++=s[i]; break;\n\t\t}\n\t\t*p=0;\n\n\treturn n;\n}\n", t, a(t));



}

int main()
{
 strcpy(t, "#include <stdio.h>\n#include <string.h>\n#include <stdlib.h>\n\nchar* a(char* s);\n\nchar t[65536];\
n\nvoid b()\n{\n\tprintf(\"%s\\n\\nint main()\\n{\\n\\tstrcpy(t, \\\"%s\\\");\\n\\tb();\\n\\treturn 0;\\n}\\n\\n
char* a(char* s)\\n{\\n\\tint l=strlen(s);\\n\\tchar* n=malloc(l*2+1);\\n\\tchar* p=n;\\n\\tfor(int i=0; i<l; i+
+)\\n\\t\\tswitch(s[i])\\n\\t\\t{\\n\\t\\t\\tcase '\\\\n':\\n\\t\\t\\t\\t*p++='\\\\\\\\'; *p++='n'; break;\\n\\t
\\t\\tcase '\\\\t':\\n\\t\\t\\t\\t*p++='\\\\\\\\'; *p++='t'; break;\\n\\t\\t\\tcase '\\\\\\\\':\\n\\t\\t\\t\\t*p
++='\\\\\\\\'; *p++='\\\\\\\\'; break;\\n\\t\\t\\tcase '\\\\\\\"':\\n\\t\\t\\t\\t*p++='\\\\\\\\'; *p++='\\\\\\\"
'; break;\\n\\t\\t\\tdefault:\\n\\t\\t\\t\\t*p++=s[i]; break;\\n\\t\\t}\\n\\t\\t*p=0;\\n\\n\\treturn n;\\n}\\n\"
, t, a(t));\n}");
 b();
 return 0;
}

char* a(char* s)
{
 int l=strlen(s);
 char* n=(char *)malloc(l*2+1);
 char* p=n;
 for(int i=0; i<l; i++)
  switch(s[i])
  {
   case '\n':
    *p++='\\'; *p++='n'; break;
   case '\t':
    *p++='\\'; *p++='t'; break;
   case '\\':
    *p++='\\'; *p++='\\'; break;
   case '\"':
    *p++='\\'; *p++='\"'; break;
   default:
    *p++=s[i]; break;
  }
  *p=0;

 return n;
}
```

运行了一下发现把这个程序自己的源代码打印了出来，同时发现这个源代码里夹杂了密文

```
11  t}\n\t\t*p=0;\n\n\treturn n;\n}\n", t, a(t));
12
13
14
15
16  '; *p++='t'; break;\n\t\t\tcase '\\\\\\\\':\n\t\t\t\t\t*p++='\\\\\\\\'; *p++='\\\\\\\\'; break;\n\t\t\tcase '\\\\\\"':\n\t
17
```

tab对应1，空格对应0

```
import re
c=''
m=re.sub(' ','0',c)
m=re.sub('\t','1',m)
#print(m)
flag=''
for i in range(0,len(c),8):
    flag+=chr(int(m[i:i+8],2))
print(flag)
```

得到flag

```
flag{D1mEn5i0nAl_Pr061em}
```