

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](#)

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/m0_46296905/article/details/121639142

版权



[CTF-MISC 专栏收录该内容](#)

4 篇文章 0 订阅

订阅专栏

文章目录

一、MISC

- (一) ISO1995
- (二) BlueTeaming
- (三) EzTime
- (四) Cipheman
- (五) threebody

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: fantasticqw2021

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:af8A04^iCT_+ $W3cz(LO)L_-s8'_<Ic/
KFP9vrr~6\ni{~#g5cs#7z2s++Y1BbYQV' iS1=DZ__|3T1QxWEwX}NJ@_3SdKK]91b?s-rS6gQwBs@4#5IGxW#&ArDw~_x"!_I^O`x5o'.s5)+c9
RU'/%_b[rjiOP0y!&/)WjKR#IjWh0\,Dr!@PH^Nf%,YoWEJ(2wXj/u~Y@gh%&_Gz5U`A=0pAV$E/ >1Kg:@4tS:V4ZB`1_x*.17B&:<xn0rW|2TY
_DSN<zvbKcj7+6w'r}Lo8':fYC@FvJ02Vb0)noQ1MI3#AZ+U3##P|W{V>z,G5[s r|Ra~-P'>oYJ{O{B"oh=VG*SB2KoNt1 7|,*r#d!=N'%0$1
Tron:7}6C(yVSIzmtw8Xza]6D,nn*q&KHnk,PW .b<h E$){Kw_)h,=m41LAv'f61: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 =}jTKX&^_Fv1b,DezJ1N}6q)76a]
Us=\u8tY;t*#}zSGo`-h64=u2bGZ)I(&%K68!nQke&+gX=L4TmMy$5nHC&+#<486HKF4f0d%1?I:1=M[p~DxBltCKh\>4<Qf+cj?a3p0F`4*-%%
7*~'^+KkQ<*z9oUgng0$:NC.Di<.`$`s+69Pn7:IgO`^T%n |Q'G&9Tx-@!6W<VK_5tH/#i>$7SKKH[Dki-o{b{?j?4.Zw+aV!|Zi{2oTqk*#!0
0h$-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，那就取中间的2个字段。

模板结果 - ISO.bt

名称	值
struct DirectoryRecord_dir	2155-255-255T255:255:00Z+02 1 B flag
ubyte ExtendedAttributeRecordLength	0
> struct uint32_LE_BE LocationOfExtent	77
> struct uint32_LE_BE DataLength	1
> Recording Data and Time	2155-255-255T255:255:00Z+02
byte hidden : 1	0

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

```

0x000000013d8af2d0 18 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Microsoft-Windows-Kernel-Power%4Thermal-Operational.evtx
0x000000013d9445e0 18 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Media Center.evtx
0x000000013d9a5d70 32 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Application.evtx
0x000000013d9a7640 33 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Windows PowerShell.evtx
0x000000013d9aae80 25 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\System.evtx
0x000000013d9abd70 32 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Security.evtx
0x000000013d9fedd0 18 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\HardwareEvents.evtx
0x000000013d9fef20 18 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Internet Explorer.evtx
0x000000013dac7070 19 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Microsoft-Windows-OfflineFiles%4Operational.evtx
0x000000013dd59530 19 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Microsoft-Windows-WindowsBackup%4ActionCenter.evtx
0x000000013e2497f0 18 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Key Management Service.evtx
0x000000013e2e8e0 1 1 RW-r-- \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Microsoft-Windows-ReliabilityAndActionCenter%4Operational.evtx

```

```
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=./
```

```

E:\CTF_project\BMZCTF\【2021强网杯】BlueTeaming>volatility -f memory.dmp --profile=Win7SP1x64 dumpfiles -Q 0x000000013d9a7640 --dump-dir=./
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x13d9a7640 None \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Windows PowerShell.evtx
SharedCacheMap 0x13d9a7640 None \Device\HarddiskVolume1\Windows\System32\winevt\Logs\Windows PowerShell.evtx

```

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

常规 详细信息

```
HostVersion=5.1.14409.1005
HostId=34c23137-ce70-41da-b0b4-20c45ba8b4e1
HostApplication=powershell -noprofile & ( $veRBOsepReFErEncE.tOstrINg()[1,3]+'x'-JOIN")( nEW-
ObjEcT sySTEm.iO.sTreaMReAdER( ( nEW-ObjEcT SystEm.iO.CompreSsiOn.DEfLATEstREam
([IO.meMoryStream] [CoNvERT]::fROMbASe64StRinG('NVJdb5tAEHyv1P9wQpYAuZDaTpvEVqRi+
5Sgmo/Axa0VRdoLXBMUmyMGU7Es//fuQvoAN7e7Nzua3RqUcJbgQVLJ1hzNi/eGLMYe2gOFX+
0zHpl9s0Uv4YHbnu8Czwl8nlW5UX4bNqM2RPGUtU4sPQSH+mmsFbIY87kFit3A6ohVnGIFbLOdLIXCdFhAIOT3rG
AEJYQvflsgmAjw/mJXTPLssxsg3U59VTvyrT7JjvDS8bwN8NvbPYt81amMeltpi1TI3omaErK0fO5bNr7LQVkwjYkqI
ZtkVtRUK8xxAQxxqylGVwM3dFX6jtw6TgbnrPRCMFlm75i3xAphq2aqUnNKfYWqhNiu0bC4wV6kXHDsh6yF5k8Xg
z7Hbi6
+ACXI/vLQyoSv7x5/EgNbXvy+VPvOAtyvWuggvuGvOhZaNFS/wTlqN9xwqGuwQddst7Rh3AfvQKHLAoCsq4jmMJ
BgKrpMbm/By8pcDQLzljju3zFn6S12zB6PjXsIfcj0XBmu8Qyqma4ETw2rd8w2MI92IGKU0HGqEGYacp7/Z2U+CB7
```

日志名称(M): Windows PowerShell
来源(S): PowerShell (PowerShell) 记录时间(D): 2020/11/26 21:00:06
事件 ID(E): 403 任务类别(Y): 引擎生命周期
级别(L): 信息 关键字(K): 经典
用户(U): 暂缺 计算机(R): WIN-T3316VDTD1I
操作代码(O):
更多信息(I): [事件日志联机帮助](#)

复制(P) 关闭(C)

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。

起始页 memory.dmp X

编辑方式: 十六进制(H) 运行脚本 运行模板

地址	十六进制	ASCII
5C76:5070h:	76 20 63 6F 64 65 20 5E 7C 20 66 69 6E 64 20 2F	v code ^ find /
5C76:5080h:	69 20 22 52 45 47 5F 53 5A 22 27 29 20 64 6F 20	i "REG_SZ"') do
5C76:5090h:	28 0D 0A 20 20 20 20 73 65 74 20 76 61 72 3D 22	(.._set var="
5C76:50A0h:	25 25 7E 61 22 3B 0D 0A 20 20 20 20 70 6F 77 65	%~a";.. powe
5C76:50B0h:	72 73 68 65 6C 6C 20 2D 6E 6F 70 72 6F 66 69 6C	rshell -noprofil
5C76:50C0h:	65 20 22 25 76 61 72 3A 7E 31 39 2C 31 35 30 30	e "%var:~19,1500
5C76:50D0h:	25 3B 0D 0A 29 0D 0A 66 6F 72 20 2F 66 20 22 74	%;..)..for /f "t
5C76:50E0h:	6F 6B 65 6E 73 3D 2A 22 20 25 25 61 20 69 6E 20	okens=*" %%a in
5C76:50F0h:	28 27 72 65 67 20 71 75 65 72 79 20 22 48 4B 45	('reg query "HKE
5C76:5100h:	59 5F 4C 4F 43 41 4C 5F 4D 41 43 48 49 4E 45 5C	Y_LOCAL_MACHINE\
5C76:5110h:	53 4F 46 54 57 41 52 45 5C 4D 69 63 72 6F 73 6F	SOFTWARE\Microso
5C76:5120h:	66 74 5C 57 69 6E 64 6F 77 73 5C 43 6F 6D 6D 75	ft\Windows\Commu
5C76:5130h:	6E 69 63 61 74 69 6F 6E 22 20 2F 76 20 63 6F 64	nication" /v cod
5C76:5140h:	65 20 5E 7C 20 66 69 6E 64 20 2F 69 20 22 52 45	e ^ find /i "RE
5C76:5150h:	47 5F 53 5A 22 27 29 20 64 6F 20 28 0D 0A 20 20	G_SZ"') do (..
5C76:5160h:	20 20 73 65 74 20 76 61 72 3D 22 25 25 7E 61 22	_set var="%~a"
5C76:5170h:	3B 0D 0A 20 20 20 20 70 6F 77 65 72 73 68 65 6C	;.. powershell
5C76:5180h:	6C 20 2D 6E 6F 70 72 6F 66 69 6C 65 20 22 25 76	l -noprofile "%v
5C76:5190h:	61 72 3A 7E 31 39 2C 31 35 30 30 25 3B 0D 0A 29	ar:~19,1500%;..)
5C76:51A0h:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5C76:51B0h:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

查找结果

地址	值
已找到 4 个 'powershell -noprofile'.	
5C7650ACh	powershell -noprofile
5C765177h	powershell -noprofile
12716125Ch	powershell -noprofile
127161327h	powershell -noprofile

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

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) 20.2 SR-5 x86

\$MFT

文件名称	描述	文件	文件大小	创建时间	修改时间	记录更新时间
{3C81A50C-1D74-400C-9264-B1F132EC7FF8}.png	存在的	png	89.9 KB	2021/05/23d00:28:33 +8	2021/05/20d12:57:23 +8	2021/05/20d12:57:23 +8
{40B9AE8-237B-499E-BB0A-FD004DC6BC52}.png	存在的	png	58.2 KB	2021/05/23d00:28:33 +8	2021/05/22d21:35:48 +8	2021/05/22d21:35:48 +8
{4101665B-4D27-4EED-BC76-27768650734F}.png	存在的	png	38.4 KB	2021/05/23d00:28:33 +8	2021/05/21d01:55:06 +8	2021/05/21d01:55:06 +8
{420A908B-16E9-4DE6-8396-DE38E0A6356C}.png	存在的	png	16.7 KB	2021/05/23d00:28:34 +8	2021/05/19d21:15:50 +8	2021/05/19d21:15:50 +8
{43726F7D-E074-468B-B7E2-9B694BB2C487}.png	存在的	png	408 KB	2021/05/23d00:28:34 +8	2021/05/20d18:52:03 +8	2021/05/20d18:52:03 +8
{450D3643-6981-4E93-8E00-E2F6AA2FE0D5}.png	存在的	png	61.0 KB	2021/05/23d00:28:34 +8	2021/05/21d02:00:01 +8	2021/05/21d02:00:01 +8
{45EF6FFC-F0B6-4000-A7C0-8D1549355A8C}.png	存在的	png	14.0 KB	2021/05/23d00:28:34 +8	2021/05/19d23:59:00 +8	2021/05/23d00:32:48 +8
{46C3DEBC-E4AD-4467-97CB-E2E2567EA349}.png	存在的	png	182 KB	2021/05/23d00:28:34 +8	2021/05/22d21:33:12 +8	2021/05/22d21:33:12 +8
{4910AC89-5F13-40C7-A474-68A2448B0CD0}.png	存在的	png	7.9 KB	2021/05/23d00:28:34 +8	2021/05/20d20:34:43 +8	2021/05/20d20:34:43 +8
{4A8E156D-8A02-48C2-BC3B-E4F6DB342BF7}.png	存在的	png	27.2 KB	2021/05/23d00:28:34 +8	2021/05/21d23:51:42 +8	2021/05/21d23:51:42 +8
{4DC658FB-6730-4934-A402-3218AAC6D3F6}.png	存在的	png	50.5 KB	2021/05/23d00:28:34 +8	2021/05/21d22:44:40 +8	2021/05/21d22:44:40 +8
{4EFA92D1-66B3-428E-9BC3-4987C0C34F76}.png	存在的	png	23.4 KB	2021/05/23d00:28:34 +8	2021/05/22d18:42:19 +8	2021/05/22d18:42:19 +8
{51B94EE3-0650-4162-B6CD-F2147F01985B}.png	存在的	png	129 KB	2021/05/23d00:28:34 +8	2021/05/19d13:39:56 +8	2021/05/19d13:39:56 +8
{5373344F-55E4-45B8-904D-1B9CE93ABC91}.png	存在的	png	24.4 KB	2021/05/23d00:28:34 +8	2021/05/19d22:05:37 +8	2021/05/19d22:05:37 +8
{53A05262-BDDF-4668-B5AA-1DAD1FCA17AE}.png	存在的	png	8.6 KB	2021/05/23d00:28:34 +8	2021/05/19d20:54:31 +8	2021/05/19d20:54:31 +8

总选中: 1 文件 (14.0 KB)

Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F

00025000 49 4C 45 30 00 03 00 FD E2 22 00 00 00 00 00

00025010 01 00 01 00 38 01 01 00 98 01 00 00 04 00 00

00025020 00 00 00 00 00 00 03 00 03 00 00 94 00 00 00

00025030 03 00 00 00 00 00 00 00 10 00 00 00 60 00 00

00025040 00 00 00 00 00 00 00 00 48 00 00 00 18 00 00

00025050 05 5A 82 27 4F D7 01 00 3A BA E1 C7 4C D7 01

00025060 20 2F 04 23 28 4F D7 01 00 3A BA E1 C7 4C D7 01

ANSI ASCII

FILEO yá"

8 -

"

#: °áçLx

ez, '0x : °áçLx

只读模式

可见磁盘空间中的分配表

簇号:

得到flag

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

(四) Cipheman

题目描述: The attacker maliciously accessed the user's PC and encrypted specific volumes. How to decrypt 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

```
0x00000007e0231c8 8 0 R--r-- \Device\HarddiskVolume2\Windows\System32\catroot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-RemoteAssistance-Pack
lient\31bf3856ad364e35\x86~6.1.7600.16385.cat
0x00000007e023810 4 0 R--r-d \Device\HarddiskVolume2\Windows\System32\drivers\rasl2tp.sys
0x00000007e023b38 8 0 R--r-- \Device\HarddiskVolume2\Windows\System32\catroot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-RemoteAssistance-Pack
lient\31bf3856ad364e35\x86-ko-KR\6.1.7601.17514.cat
0x00000007e025468 8 0 R--r-- \Device\HarddiskVolume2\Windows\System32\catroot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Branding-HomeBasic-Cl
Package\31bf3856ad364e35\x86~6.1.7600.16385.cat
0x00000007e025db0 8 0 R--r-- \Device\HarddiskVolume2\Windows\System32\catroot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Branding-HomeBasic-Cl
Package\31bf3856ad364e35\x86-ko-KR\6.1.7601.17514.cat
0x00000007e02aa88 6 0 R--r-d \Device\HarddiskVolume2\Windows\System32\msacm32.dll
0x00000007e02abd0 6 0 R--r-d \Device\HarddiskVolume2\Windows\System32\midimap.dll
0x00000007e02af80 8 0 -w---- \Device\HarddiskVolume2\Users\RockAndRoll\Desktop\BitLocker 氩店憐 斌?168F1291-82C1-4BF2-B634-9CCCEC63E9ED.txt
Traceback (most recent call last):
  File "vol.py", line 192, in <module>
  File "vol.py", line 183, in main
```

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 0x00000007e02af80 --dump-dir=./
```

得到内容如下

```
$MFT $LogFile 起始页 memory file: None. 0x86b45d78.txt x
编辑方式: Unicode(U) 运行脚本 语法
BitLocker 드라이브 암호화 복구 키
□복구 키는 BitLocker로 보호되는 드라이브에서 데이터를 검색하기 위해 사용됩니다.
이 키가 올바른 복구 키인지 확인하려면 복구 화면에 표시된 것과 ID를 비교하십시오.
복구 키 ID: 168F1291-82C1-4B
전체 복구 키 ID: 168F1291-82C1-4BF2-B634-9CCCEC63E9ED
BitLocker 복구 키:
221628-533357-667392-449185-516428-718443-190674-375100
```

BitLocker 드라이브 암호화 복구 키
복구 키는 BitLocker로 보호되는 드라이브에서 데이터를 검색하기 위해 사용됩니다
이 키가 올바른 복구 키인지 확인하려면 복구 화면에 표시된 것과 ID를 비교하십시오.
BitLocker 복구 키:
221628-533357-667392-449185-516428-718443-190674-375100

BitLocker驱动器加密恢复密钥
恢复密钥将被用于在BitLocker保护的驱动器中搜索数据
要确认此密钥是否正确的恢复密钥, 请比较显示在恢复屏幕上的密钥和ID。
BitLocker恢复键:
221628-533357-667392-449185-516428-718443-190674-375100

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



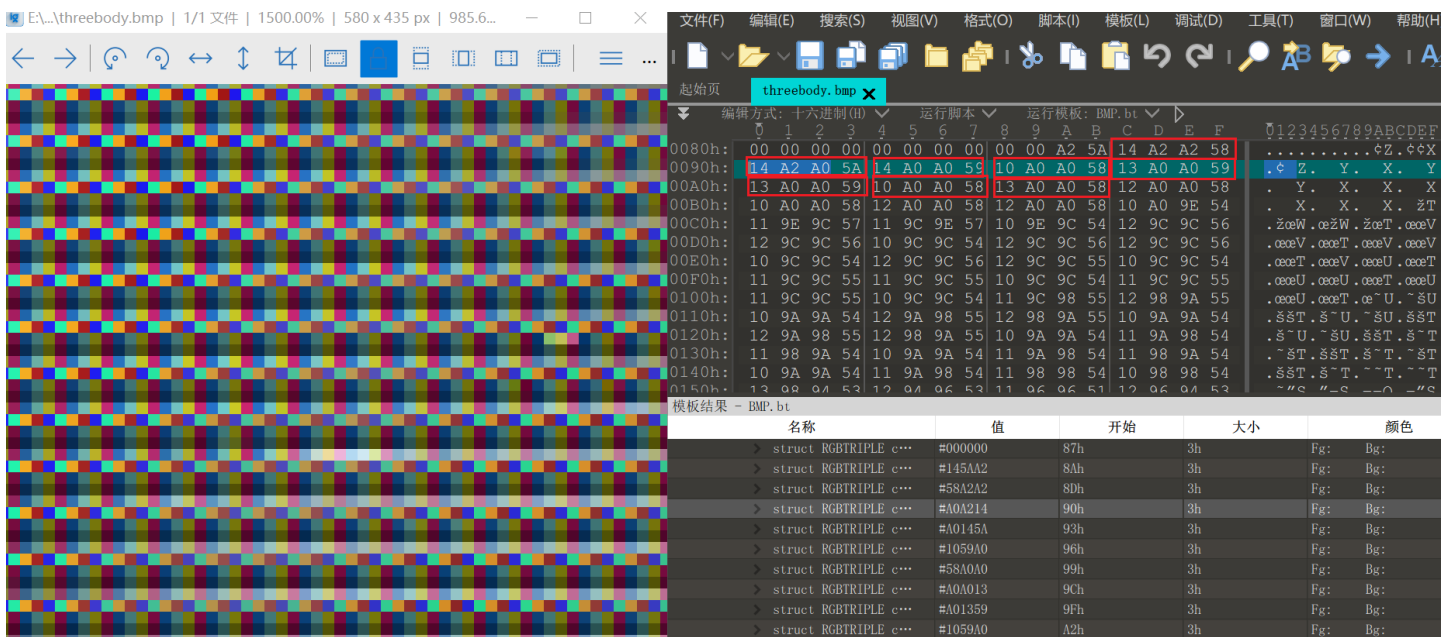
得到flag

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 - wdxg'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）



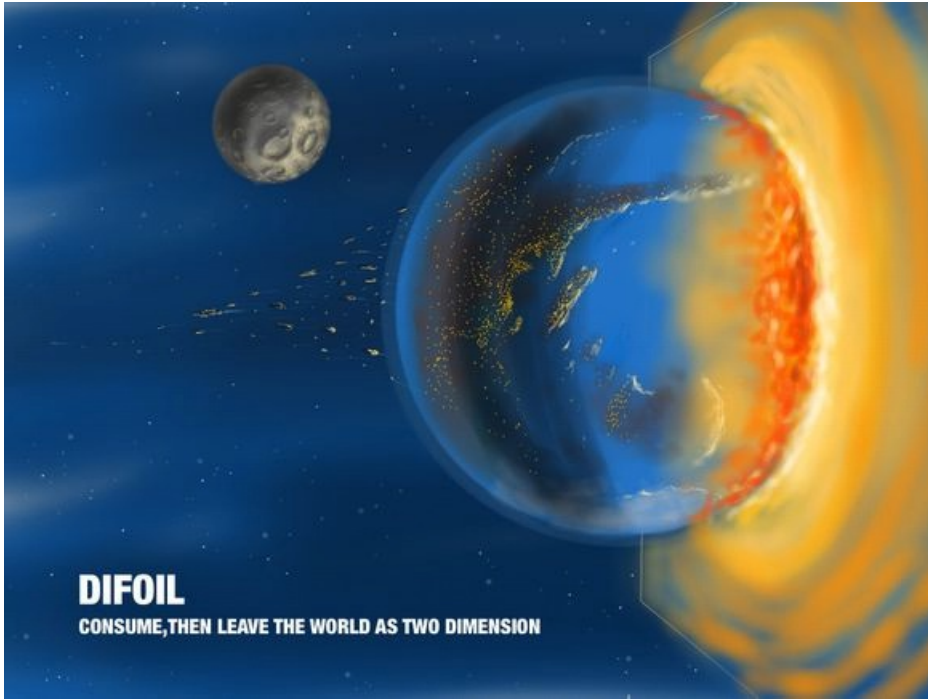
所以修改biBitCount字段值为32，变成4个字节一个像素

```
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
```

反结果 - 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

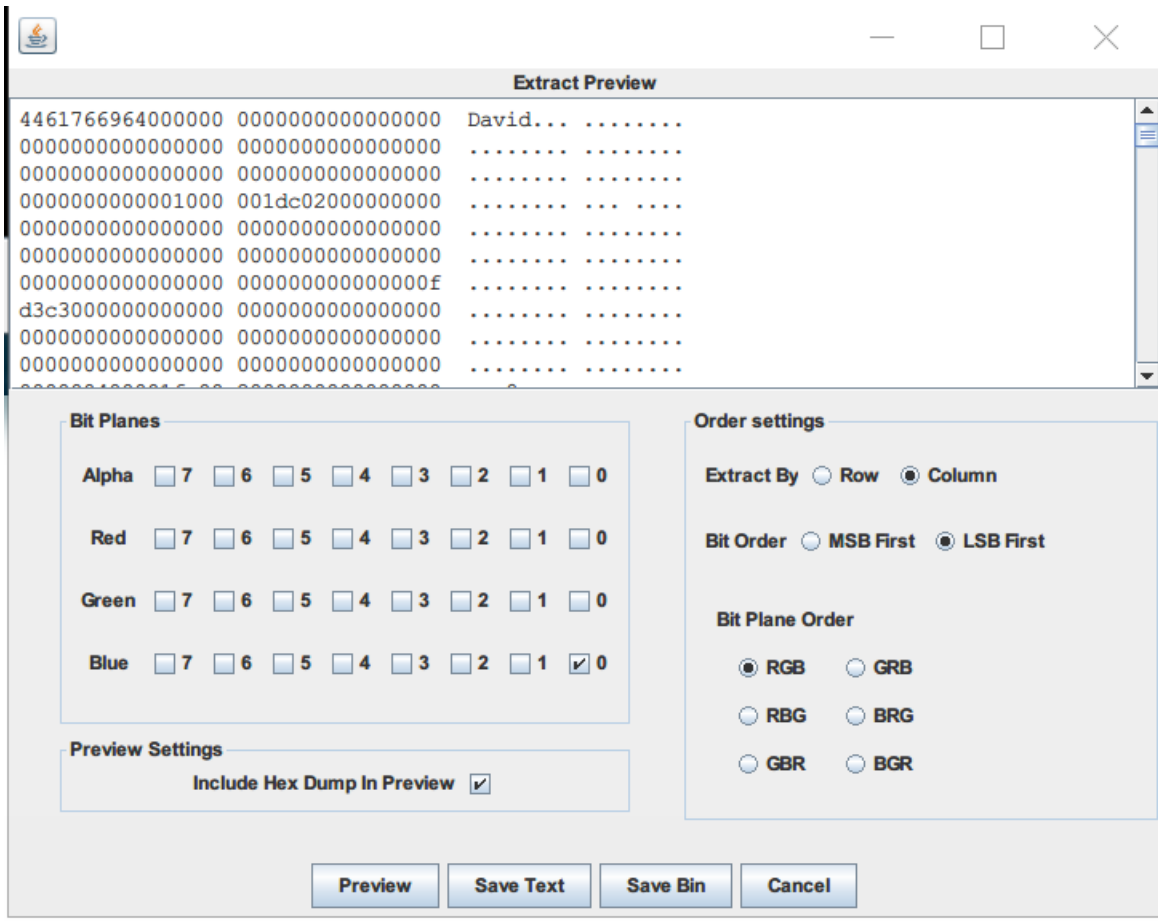
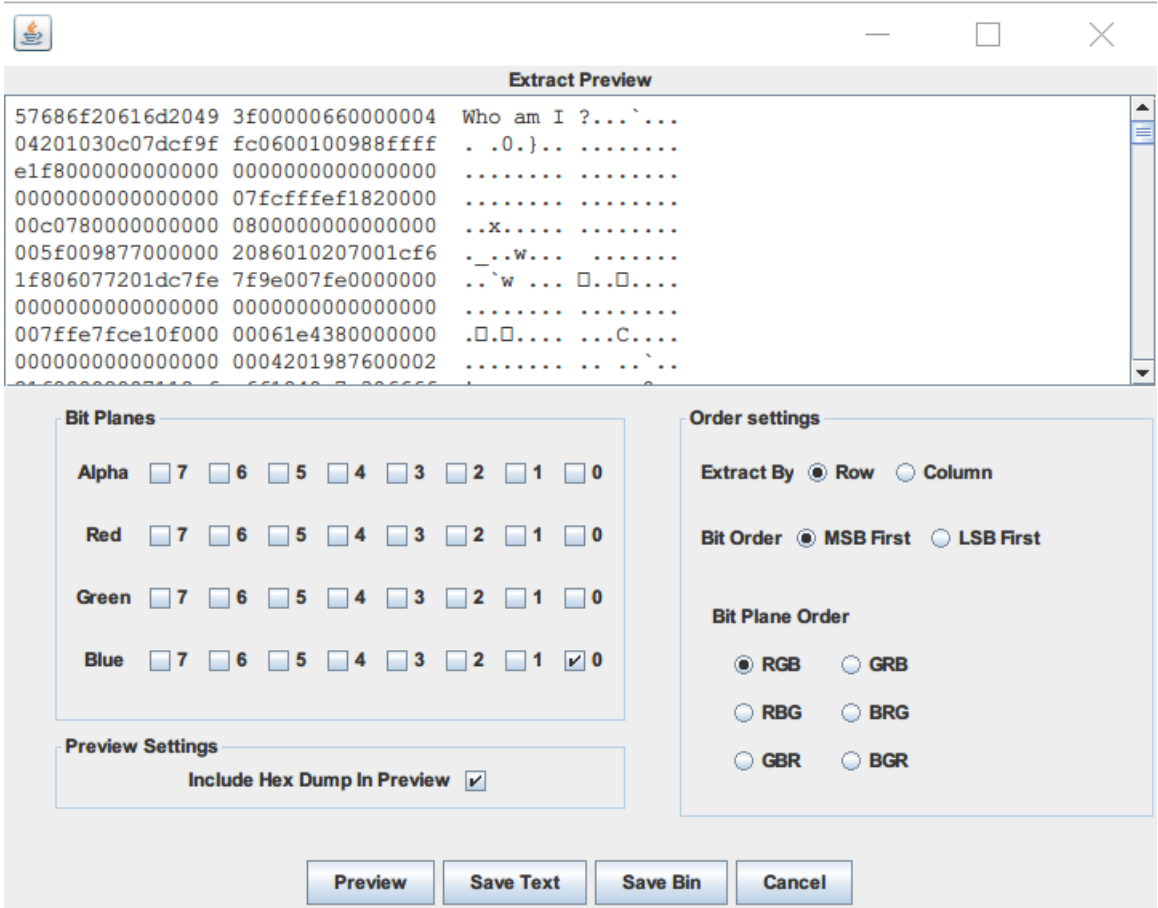
保存后得到的图片如下



StegSolve中看RGB各个通道，只有这个正常些



行列分别有LSB隐写

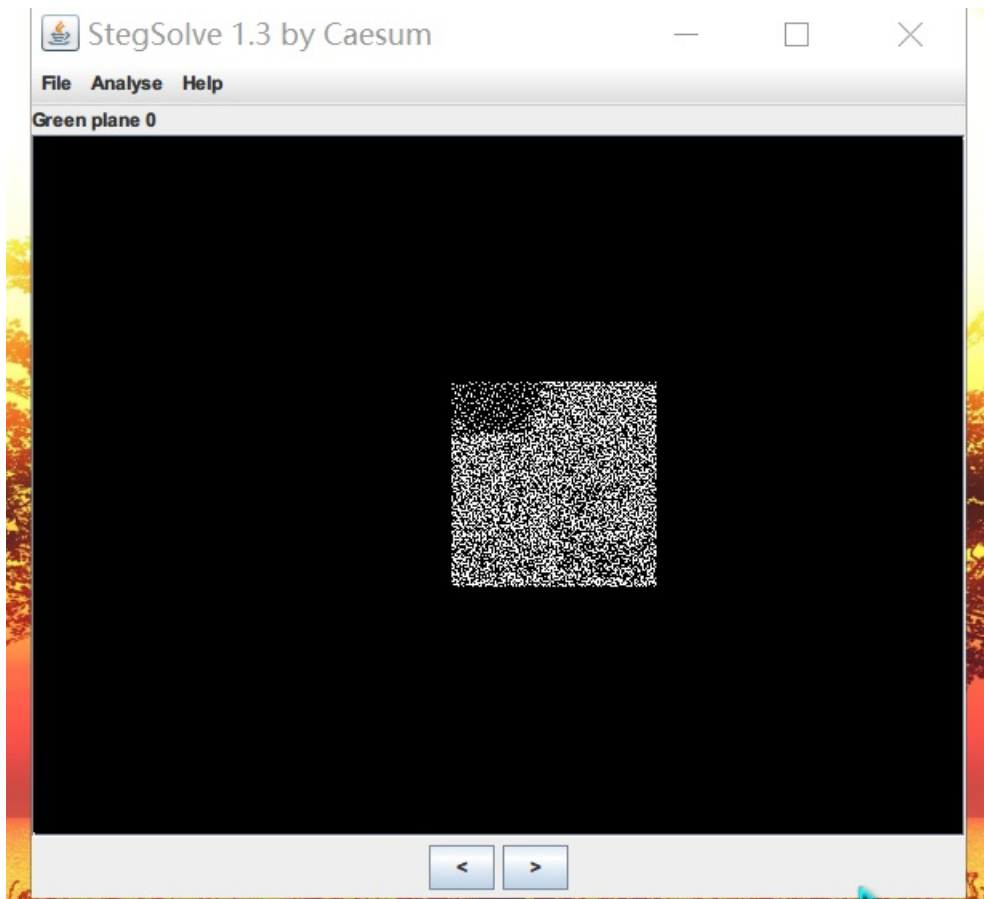


网上搜一下David，查到这个人是戴维·希尔伯特，一个数学家

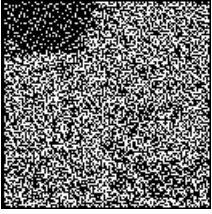
用StegSolve只能看到bmp的RGB三个通道，不能看到rgbReserved通道的情况（试过了，Alpha通道看到的是白屏），本来想找个在线工具把bmp转png，然后发现没有用，可能是那个网站转换算法有问题把，所以无奈之下写个脚本转换成png格式再去看看各个通道的情况。

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

char* a(char* s)
{
    int l=strlen(s);
    char* n=(char *)malloc(1*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

