Misc做题总结（做题一时爽）

原创

将至将至 于 2019-03-18 22:56:46 发布 21194 收藏 105

分类专栏： CTF-Misc 文章标签： CTF Misc

本文链接：https://blog.csdn.net/Laurel_60/article/details/82154712

文章目录

№.1 隐写

Stegsolve一般怎么用呢？

Winhex又怎么用呢？

最后还有一个Binwalk~

№.2 流量分析

来看几道题目：

№.3 NTFS交换数据流隐写

日常做题：

偶尔做题：

> https://laure1.github.io/2019/08/09/Summary-of-Misc/

360CTF的流量题（题目很任性，质量挺高）：

> https://blog.csdn.net/Laurel_60/article/details/102932373

2019安洵杯：

> https://blog.csdn.net/Laurel_60/article/details/103334550

№.1 隐写

> 好久没更过这篇文了，今天心情好，正好最近培训刷了一些题目，就来更一波吧嘿嘿嘿？

Misc，杂项，就是很杂的意思。

很多时候做Misc题拿到的都是一张图片（当我没说，

最简单的，记事本打开，看看数据中是否有隐藏的flag，password等东西。

最常见的，右键->其他压缩命令->360压缩打开，没安装360压缩的改文件后缀名为rar或者zip然后解压缩打开。

很常见的，Stegsolve打开，看看有木有隐藏的文字或者--二维码

必不可少的，Winhex打开，查看有木有隐藏的文件或者文件头有木有损坏。

也比较常见的，Binwalk分析，查看有木有隐藏的文件。

总而言之，以上基本上都试试吧2333333下面一个个介绍一下⑧~~

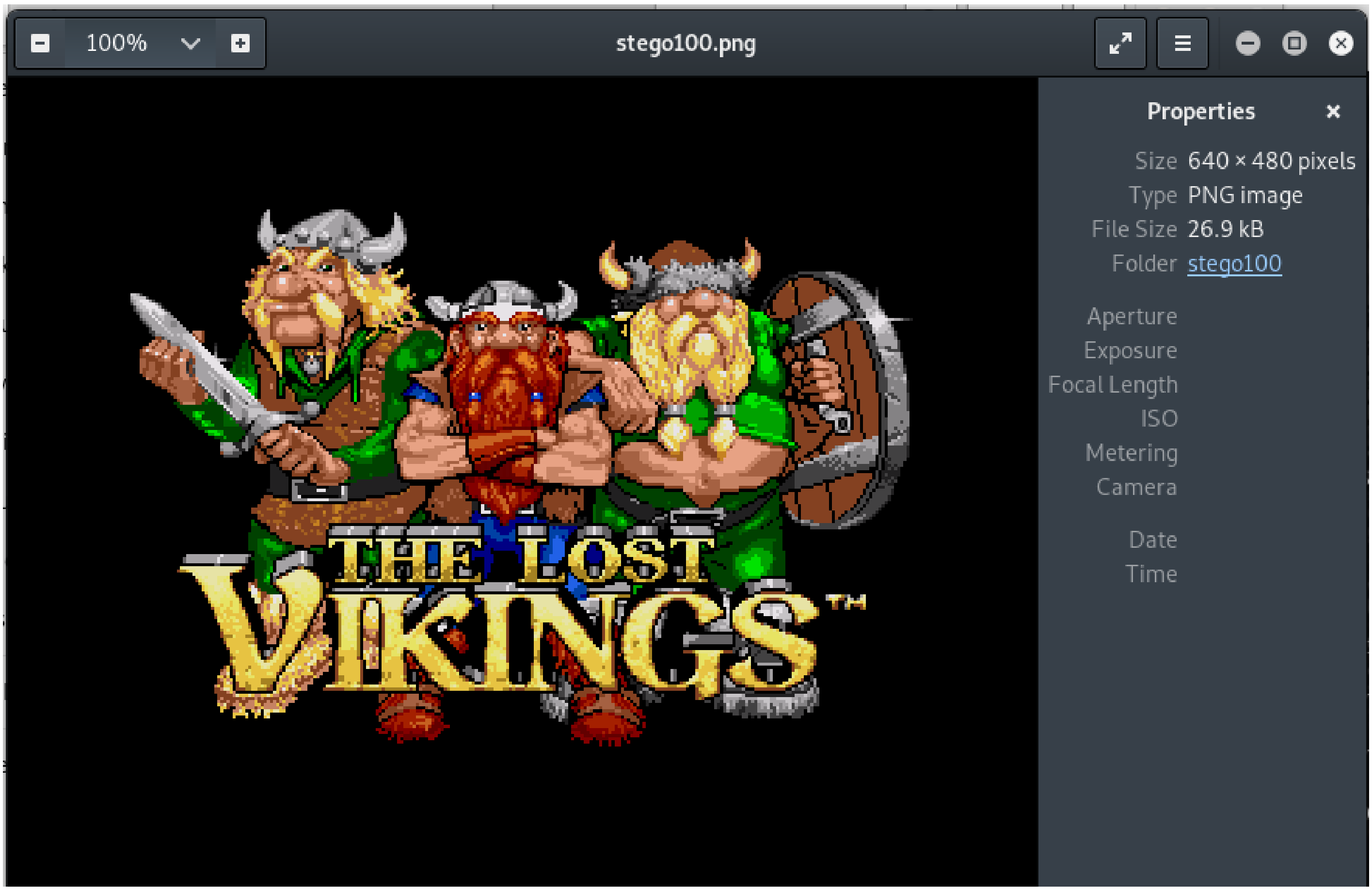
Stegsolve一般怎么用呢？

> 话说最近做题都没怎么用它了。。。

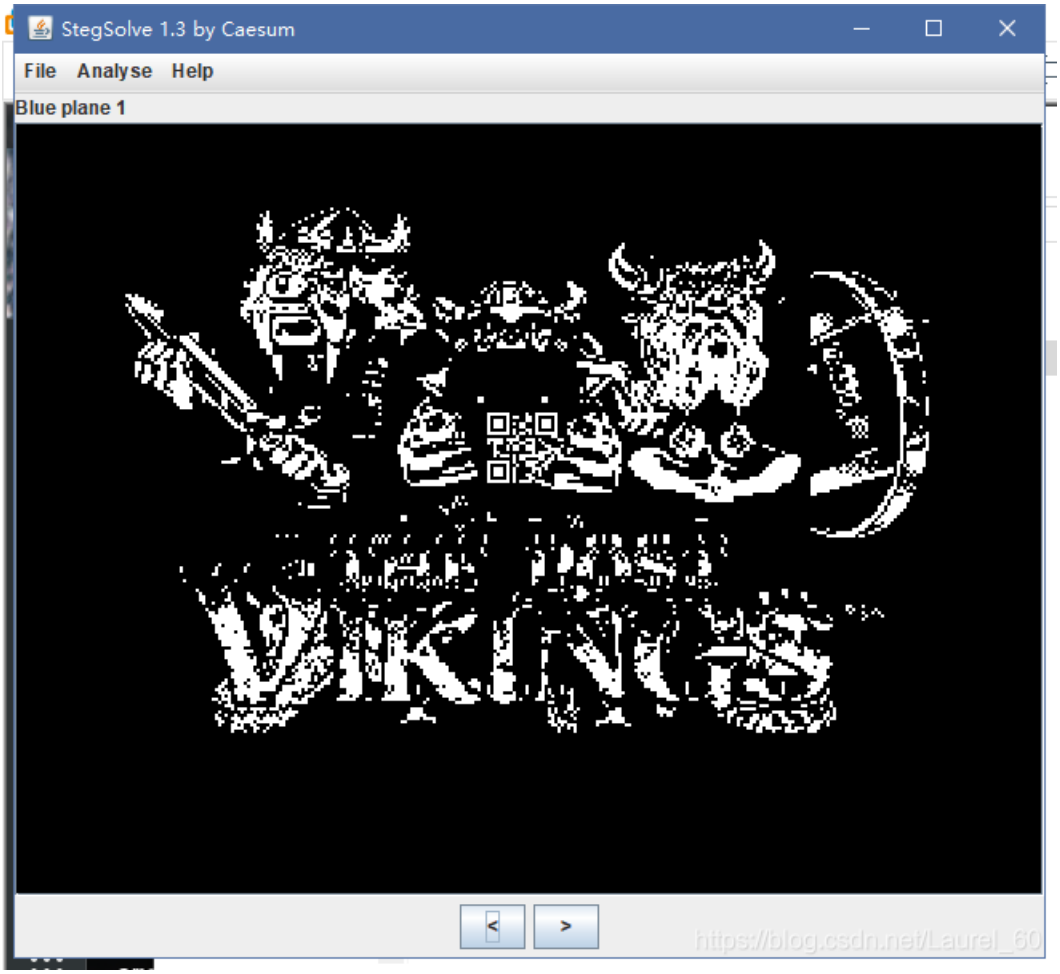
- 最基础的，左右切换颜色通道，看看会不会突然出现令人激动的东西。

,,

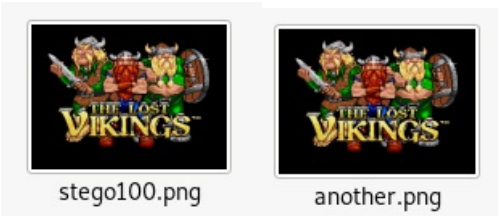
eg：题目给出了一张图片--



毕竟题目标题里面含了stego，所以首先就用stegosolve简单分析一下--



在blue plane1的时候可以看到正中间有一个二维码。想直接扫吧，但是无法识别，这就有点麻烦了。看到有人拖到了什么软件里面改通道颜色进行还原？？wkiao师傅们太猛了吧。。

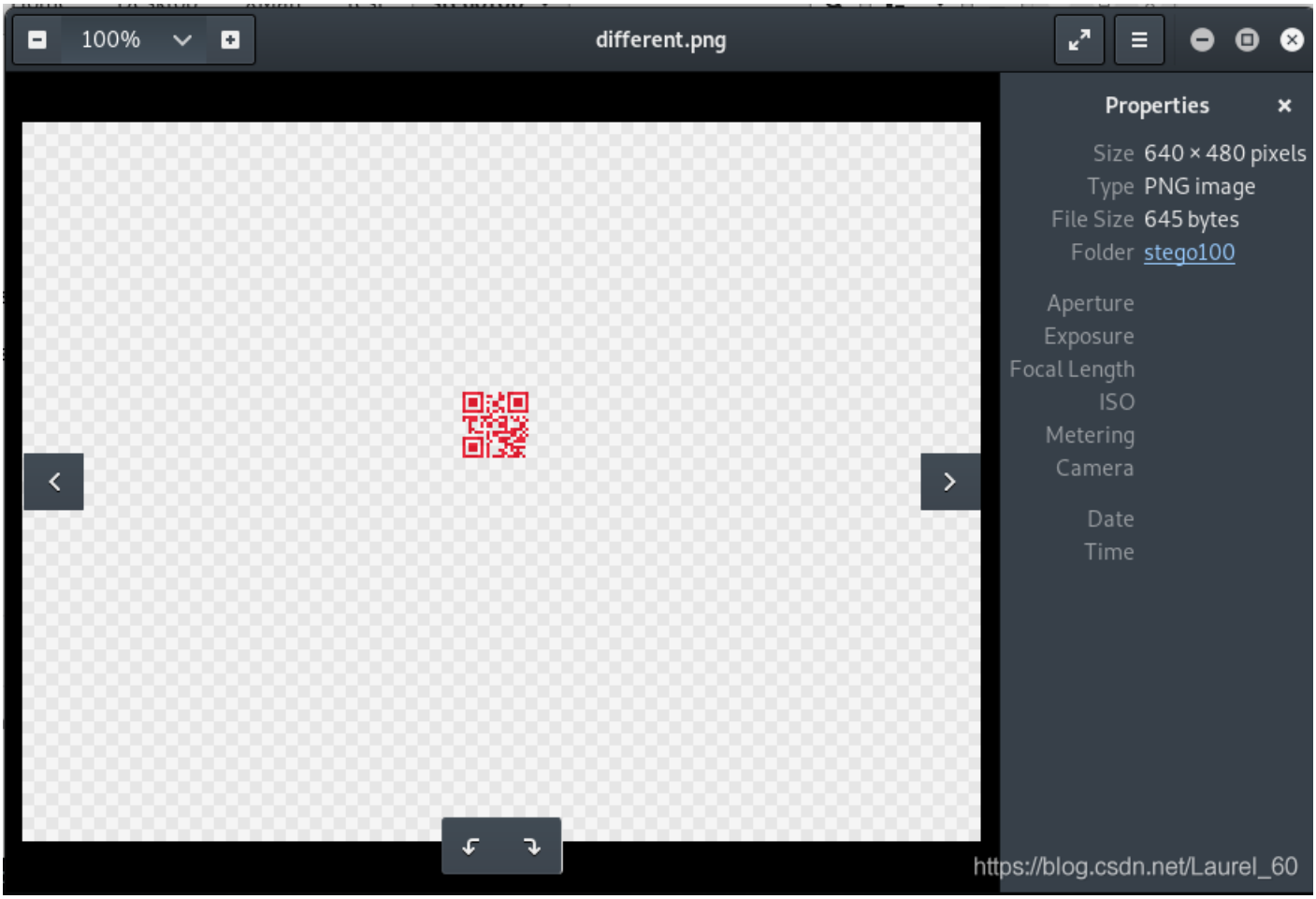


看到国外的师傅的一个解法，在网上找一张相似（几乎一模一样）的图片--

然后用命令--

```
compare stego100.png another.png -compose src different.png
```

提取出隐藏的二维码，可以直接扫的!!!--



贼好用，膜一波~~~~

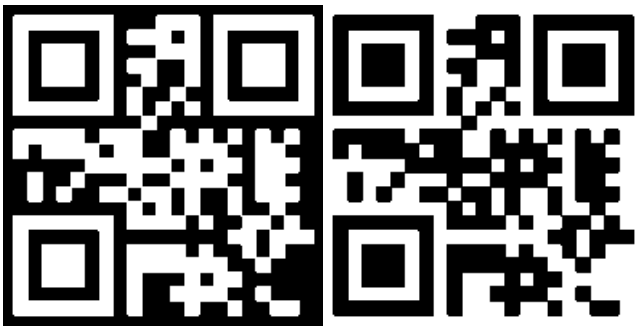
- Image Combiner

  如果题目给了两张图片，就可以用这个功能把两张图片组合在一起分析，往往能出现令人激动的东西。

‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘

eg：实验吧的“男神”那道题就是这个思路，用stegsolve这图分析一下那图分析一下组合分析一下具体咋分析我也忘了，反正最终总共出来三张二维码，当然肯定还是有坑的。

作为一个坐了好几年动车的人，都没有注意过二维码的特点，连反色了都不知道，还想着为啥死活扫不出来，也是智商为负了--



二维码都长上面第二个图那样，三个标准的方块，中间可以放个图标，不影响。所以有些时候出题人会很贴心地把某几个方块遮住，比如下面这张图就是--

用一个很可(sao)爱(qi)的狗狗成功遮挡住了重要部位，怎么解决呢，用电脑自带的画图工具，选中二维码某一个小方块，复制粘贴到下面把狗狗遮了就行了。

- Data Extract

  一般用于分析流隐写，把后三位通道点为0，LSB First，然后进行分析。

- FrameBrower

  如果是个动图，这个功能可以把这个动图分成一帧一帧的。如图片上有flag一闪而过，那还好办；如果不是，那就有可能是时间表有猫腻，试着分析一下，说不定是二进制码什么的。

Winhex又怎么用呢？

> 前两天被Winhex坑了，瞬间失去了好感?
> 不过前两天有师傅给了我她的winhex，永久的那种，好喜欢啊❤

- 分析文件头部和尾部。

如果遇到题目给的图片文件出现打不开的状况，然而又的确是个图片文件，这时可以试试用Winhex打开文件，观察文件的文件头数据是否丢失或者改写了。压缩包等文件同理。我无聊的时候还是整理了一下基本上会遇到的文件头和尾，这里直接贴出来- -

各文件的文件头和尾标识符
GIF文件头：47 49 46 38 39 61
JPG文件头：FF D8 FF Eo oo 10 4A 46 49 46
FF D8 FF E1 oo 10 4A 46 49 46
JPG文件尾：FF D9

PNG文件头：89 50 4E 47 oD oA 1A oA
PNG文件尾：00 00 00 00 49 45 4E 44 AE 42 60 82
PSD文件头：38 42 50 53
TIFF文件头：49 49 2A 00
BMP文件头：42 4D

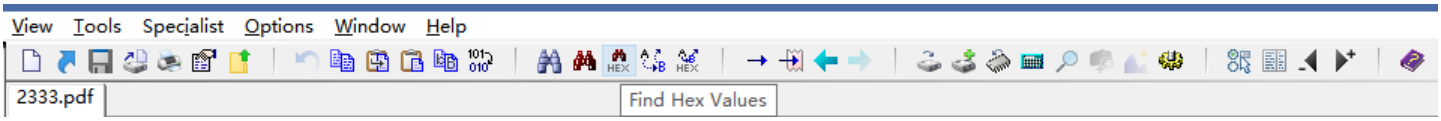
Rar文件头：52 61 72 21
zip文件头：50 4B 03 04

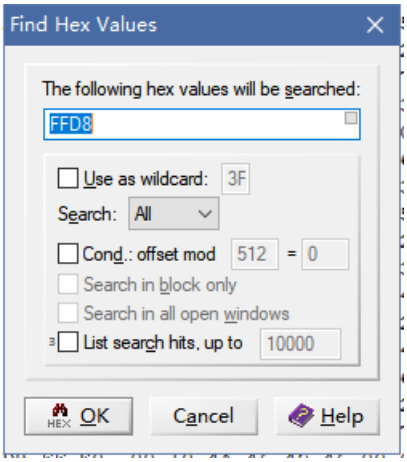
- 修改图片的高度和宽度。

有些时候可能会给你一张看起来很正常的图片，binwalk分析一下发现什么也没有藏，那就可能是图片被改“短”了。

- 搜索文件头部查看是否有隐藏的文件。

分析隐藏文件的话，可以先用binwalk直接分析，如果binwalk -e提取不出来的话就用winhex搜索或者根据地址进行提取，我个人觉得还是010Editor的搜索功能好用些，哦不，是好用很多--





曾经遇到过一道题目就是直接在搜索界面搜索RAR，可以看到一个压缩包的头部，当然这个头部是残缺的。从这个头部开始，导出后面的所有文件，然后补全头部就可以了。很久以前不知道咋导出，废了好大的力气，实际上--

❤从起始位置开始，右键->Start of block，然后拉到最后，右键->End of block，就可以把整个选块都选中了❤

最后还有一个Binwalk~

> 之前没写Binwalk，今天补上。

Binwalk是kali自带的工具(Ubuntu可能也有吧，我没试过)，当然windows也是可以装的，不过我还是喜欢用现成的鸭~毕竟我比较懒。

Binwalk的命令其实很简单--

```
binwalk 文件
```

,,,

eg：某一道题目是一个pdf文件- -

用binwalk分析一下- -

```
root@sp:~/Desktop/XMan/test# binwalk 2333.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PDF document, version: "1.4"
452           0x1C4           JPEG image data, JFIF standard 1.01
73254         0x11E26         JPEG image data, JFIF standard 1.01
81606         0x13EC6         Zlib compressed data, default compression
82150         0x140E6         JPEG image data, JFIF standard 1.01
104469        0x19815         Zlib compressed data, default compression
105134        0x19AAE         Zlib compressed data, default compression
```

发现里面有三张图片，但是在pdf文件里面只显示了两张，那第三张肯定有猫腻，说不定就是flag。所以我们需要把那三张图片提取出来(其实就提取最后一张就可以了，不过谁让我闲呢?)

如果用binwalk -e命令，是提不出来的，所以我们得用Editor提取。

于是我用了Winhex，当时差点没把我给气死。= =

从binwalk里面可以看到第三张图片的起始位置是0x140E6，那我们就找到这个位置--

```
000140B0  44 65 76 69 63 65 52 47  42 0A 2F 4C 65 6E 67 74  DeviceRGB /Lengt
000140C0  68 20 31 38 20 30 20 52  0A 2F 46 69 6C 74 65 72  h 18 0 R /Filter
000140D0  20 2F 44 43 54 44 65 63  6F 64 65 0A 3E 3E 0A 73   /DCTDecode >> s
000140E0  74 72 65 61 6D 0A FF D8  FF E0 00 10 4A 46 49 46  tream ÿØÿà  JFIF
000140F0  00 01 01 01 00 48 00 48  00 00 FF DB 00 43 00 02       H H  ÿÛ C
00014100  01 01 02 01 01 02 02 02  02 02 02 02 02 03 05 03
00014110  03 03 03 03 06 04 04 03  05 07 06 07 07 07 06 07
00014120  07 08 09 0B 09 08 08 0A  08 07 07 0A 0D 0A 0A 0B
00014130  0C 0C 0C 0C 07 09 0E 0F  0D 0C 0E 0B 0C 0C 0C FF                ÿ
00014140  DB 00 43 01 02 02 02 03  03 03 06 03 03 06 0C 08  Û C
00014150  07 08 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C
00014160  0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C
00014170  0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C
00014180  0C 0C 0C 0C FF C0 00 11  08 02 88 04 80 03 01 22      ÿÀ     ˆ €  "
00014190  00 02 11 01 03 11 01 FF  C4 00 1F 00 00 01 05 01        ÿÄ
000141A0  01 01 01 01 01 00 00 00  00 00 00 00 00 01 02 03
000141B0  04 05 06 07 08 09 0A 0B  FF C4 00 B5 10 00 02 01          ÿÄ µ
000141C0  03 03 02 04 03 05 05 04  04 00 00 01 7D 01 02 03                }
000141D0  00 04 11 05 12 21 31 41  06 13 51 61 07 22 71 14       !1A  Qa "q
```

图片尾部的位置应该在0x19815的前面，找一找，毕竟前后差距很大的，所以一眼就能看到--

```
00019620  00 A2 8A 28 00 A2 8A 28  00 A2 8A 28 00 A2 8A 28   ¢Š( ¢Š( ¢Š( ¢Š(
00019630  00 A2 8A 28 00 A2 8A 28  00 A2 8A 28 00 A2 8A 28   ¢Š( ¢Š( ¢Š( ¢Š(
00019640  00 A2 8A 28 00 A2 8A 28  00 A2 8A 28 00 A2 8A 28   ¢Š( ¢Š( ¢Š( ¢Š(
00019650  00 A2 8A 28 00 A2 8A 28  00 A2 8A 28 00 A2 8A 28   ¢Š( ¢Š( ¢Š( ¢Š(
00019660  00 A2 8A 28 00 A2 8A 28  00 A2 8A 28 00 A2 8A 28   ¢Š( ¢Š( ¢Š( ¢Š(
00019670  03 FF D9 65 6E 64 73 74  72 65 61 6D 0A 65 6E 64  ÿÙendstream end
00019680  6F 62 6A 0A 31 38 20 30  20 6F 62 6A 0A 32 31 39  obj 18 0 obj 219
00019690  30 31 0A 65 6E 64 6F 62  6A 0A 31 36 20 30 20 6F  01 endobj 16 0 o
000196A0  62 6A 0A 3C 3C 0A 2F 54  79 70 65 20 2F 50 61 67  bj << /Type /Pag
000196B0  65 0A 2F 50 61 72 65 6E  74 20 33 20 30 20 52 0A  e /Parent 3 0 R
000196C0  2F 43 6F 6E 74 65 6E 74  73 20 31 39 20 30 20 52  /Contents 19 0 R
000196D0  0A 2F 52 65 73 6F 75 72  63 65 73 20 32 31 20 30   /Resources 21 0
000196E0  20 52 0A 2F 41 6E 6E 6F  74 73 20 32 32 20 30 20   R /Annots 22 0
000196F0  52 0A 2F 4D 65 64 69 61  42 6F 78 20 5B 30 20 30  R /MediaBox [0 0
```

现在我们把这部分提取出来--

2333.pdf | noname

```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F      ANSI ASCII
00000000  FF D8 FF E0 00 10 4A 46  49 46 00 01 01 01 00 48  ÿØÿà  JFIF     H
00000010  00 48 00 00 FF DB 00 43  00 02 01 01 02 01 01 02   H  ÿÛ C
00000020  02 02 02 02 02 02 02 03  05 03 03 03 03 03 06 04
00000030  04 03 05 07 06 07 07 07  06 07 07 08 09 0B 09 08
00000040  08 0A 08 07 07 0A 0D 0A  0A 0B 0C 0C 0C 0C 07 09
00000050  0E 0F 0D 0C 0E 0B 0C 0C  0C FF DB 00 43 01 02 02         ÿÛ C
00000060  02 03 03 03 06 03 03 06  0C 08 07 08 0C 0C 0C 0C
00000070  0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C
00000080  0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C
00000090  0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C FF C0                ÿÀ
000000A0  00 11 08 02 88 04 80 03  01 22 00 02 11 01 03 11     ˆ €  "
000000B0  01 FF C4 00 1F 00 00 01  05 01 01 01 01 01 01 00  ÿÄ
000000C0  00 00 00 00 00 00 00 01  02 03 04 05 06 07 08 09
000000D0  0A 0B FF C4 00 B5 10 00  02 01 03 03 02 04 03 05    ÿÄ µ
```

保存为jpg文件之后就可以看到flag了。

№.2 流量分析

> 流量分析晚上再加吧。。。。

流量分析题考察我们对于网络流量包的分析能力，可能需要我们从中提取出一个文件，也可能flag就隐藏在数据包中。遇到该类题型，可以使用Wireshark进行分析。Wireshark的过滤器可以帮助我们迅速定位到要分析的报文。还可以将HTTP流或TCP流汇聚或还原成数据显示出来。

而pcap文件格式是常用的数据报存储格式，wireshark这类的主流抓包软件生成这种格式的数据包，格式什么的，就没仔细看过。。略过略过。。

一般来讲，我拿到pcap文件的话，
会先用过滤器->字符串查一下分组字节流有没有flag什么的，没有是正常的。
然后导一下TCP流，看看有没有flag图片啊什么的。
或者还有其他的骚操作。❤

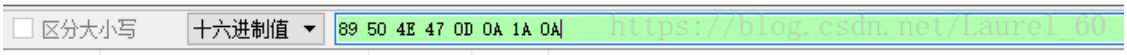
来看几道题目：

- 题目一

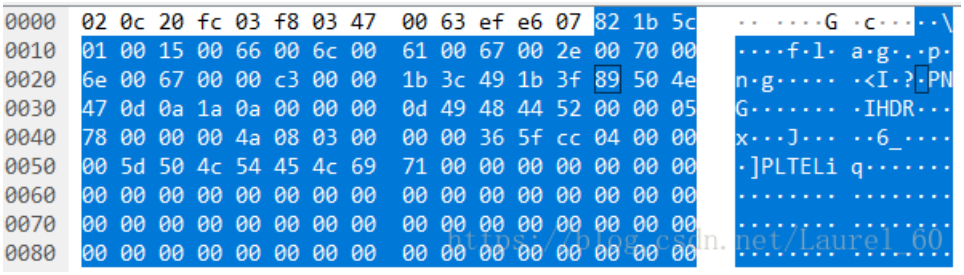
题目给了一个pcap文件，可以先用binwalk分析该文件，发现包含了一个PNG格式的文件，猜测flag应该就隐藏在该文件中。

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
40535         0x9E57          PNG image, 1400 x 74, 8-bit colormap, non-interlaced
```


然后用wireshark开始分析。这个题目是使用filter查找文件头--



可以查到有png文件头的一部分字节--



再按照Source进行排序，通过比对和观察可以发现，从1052到952开头的都是相应PNG文件的一部分，952下含有图片尾部字节。

```
0330  6c a6 dd 6a b9 ad 07 d7  09 d4 ad d1 22 f1 c0 0b   l··j···· ····"···
0340  f6 36 87 47 4d 88 7d ba  82 bb 89 b5 b5 40 dd 62   ·6·GM·}· ·····@·b
0350  2d 3a e2 81 17 96 62 57  54 f9 78 2b 3f b9 7e 98   -:····bW T·x+?·~·
0360  27 ab d5 7e 26 50 b7 48  8b c4 03 2f c8 0a e1 fd   '··~&P·H ···/····
0370  ae cd c7 2c f6 d9 b8 72  00 f0 cb a6 83 cd ca 86   ···,···r ········
0380  f6 aa aa f6 3e d6 82 03  80 df 55 c6 c6 e8 e0 b4   ····>··· ··U·····
0390  4d e7 69 5e 84 b0 70 00  f0 cb 8a 10 fc 28 99 57   M·i^··p· ·····(·W
03a0  ee a9 6b 6f 61 d1 a3 34  8e 1d 57 00 00 00 00 49   ··koa··4 ··W····I
03b0  45 4e 44 ae 42 60 82 0e                           END·B`··
```


所以只需要把这些部分下的字节拼成原文件就可以得到该png图片。

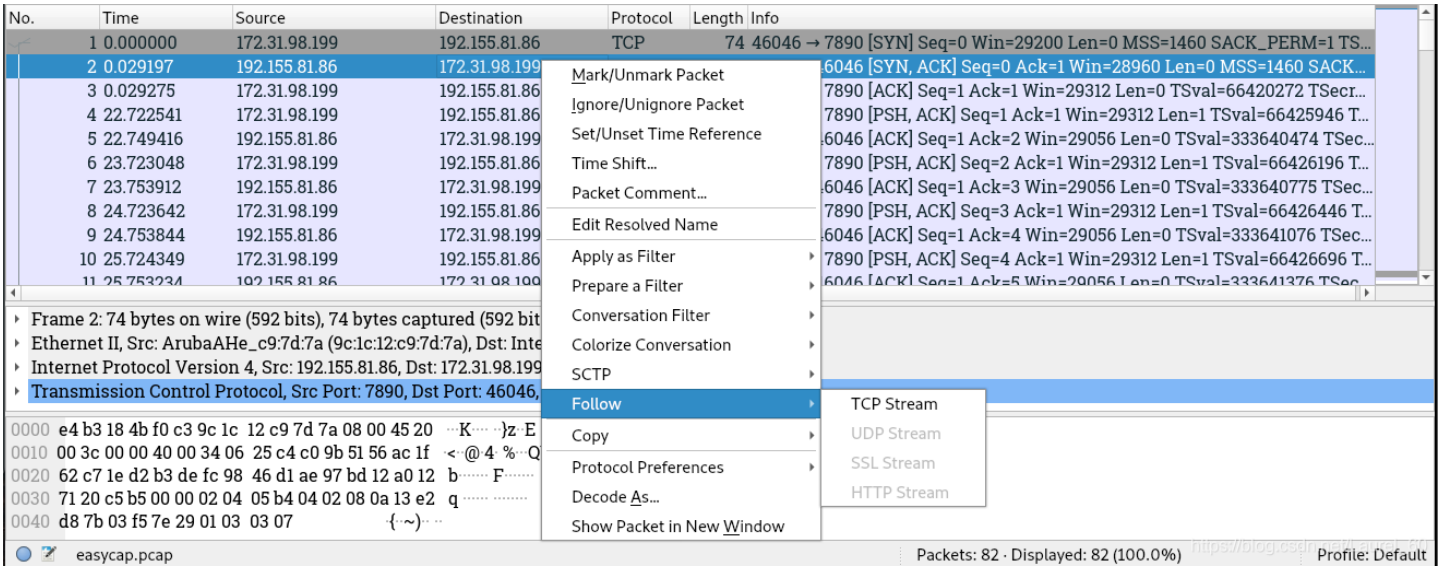
```
SamsungE_7e:e7:54 (… RFCOMM     20 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM   1025 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM   1025 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM   1025 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM   1025 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM   1025 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM   1025 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM    952 Sent UIH Channel=12
SamsungE_7e:e7:54 (… RFCOMM     16 Sent UIH Channel=12
```


> PS:此处需注意的是，在复制时和在粘贴时都有多个选项，在wireshark上应选择Copy Bytes as a Hex Stream，在winhex上粘贴时选择ASCII Hex。

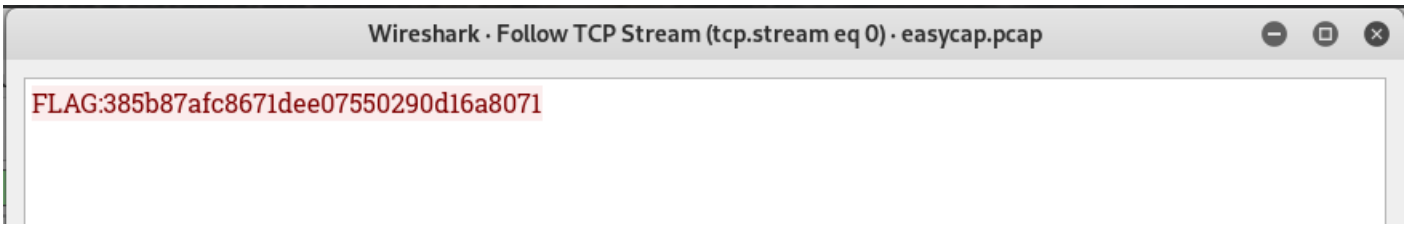
- 题目二 easypcap

名字叫easypcap，正常套路下那肯定不easy，但是这个出题人比较耿直，因为这道题是真的真的真的很easy。

打开全是TCP流，习惯性地右键追踪TCP流- -



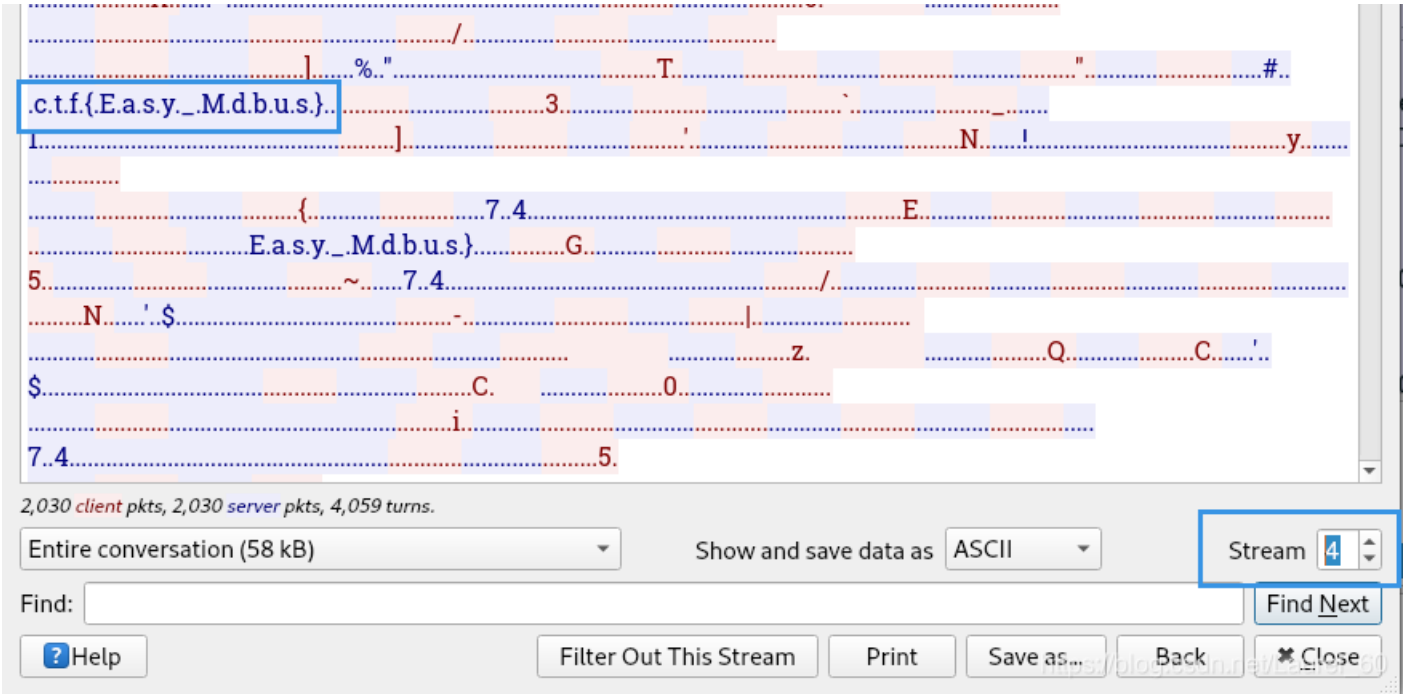
然后就有flag了- -



> 对没错这尼玛就是flag。。提交的时候我还想着要不换成花括号吧，，欸怎么不对？？该不会那一串数字还得解吧，嗯肯定不会这么简单，md5解密试试哇不行欸哇那怎么办啊哇我不会了哇。。。。嗯我傻了。。。

- 题目三 神奇的modbus

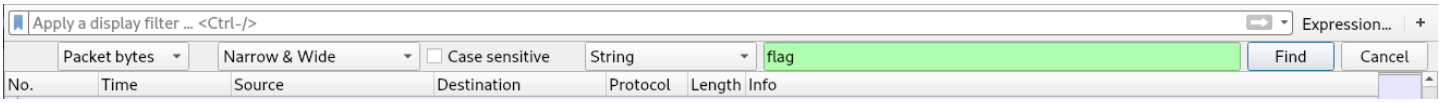
打开之后还是直接右键追踪TCP流，当然你想追踪UDP也没人拦你
然后就一个个流翻着走，看看有没有什么特别的地方。第四个流里面就有flag哦~~



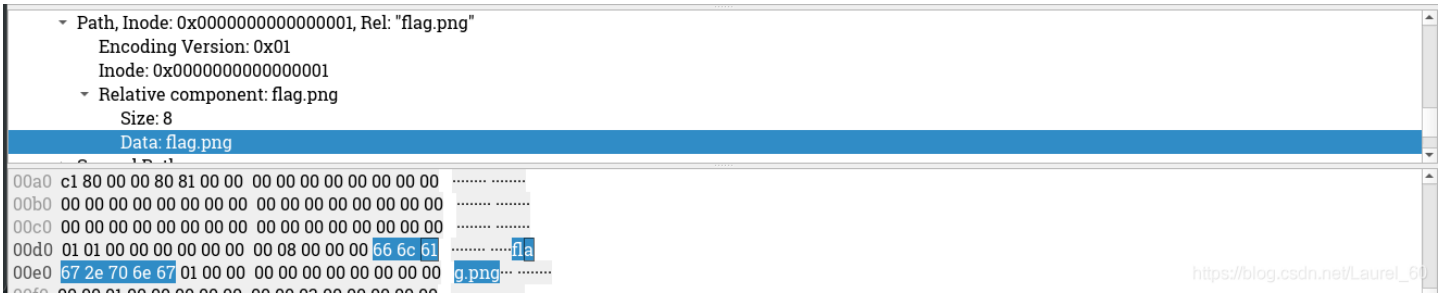
不过，这个flag里面的单词是mdbus，提交是错误的，所以需要加个o变成modbus

- 题目四

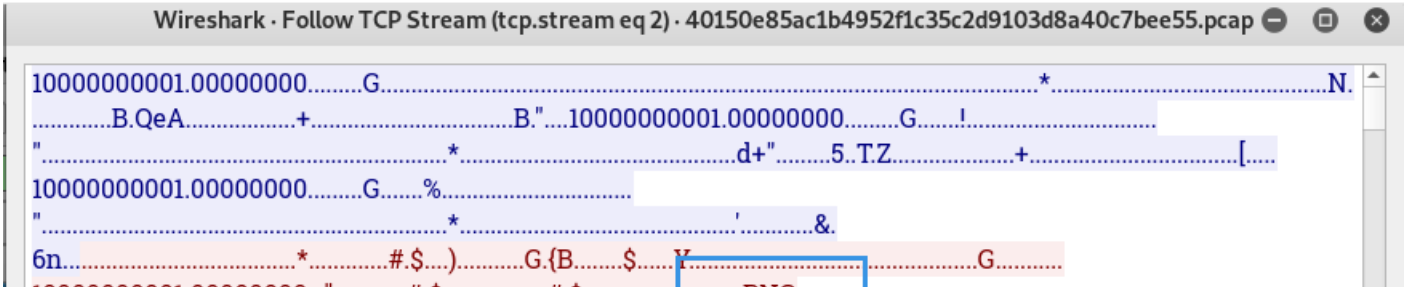
前面两道题都太简单了，好吧其实这道题也比较简单。。。
用过滤器搜索一下flag试试- -

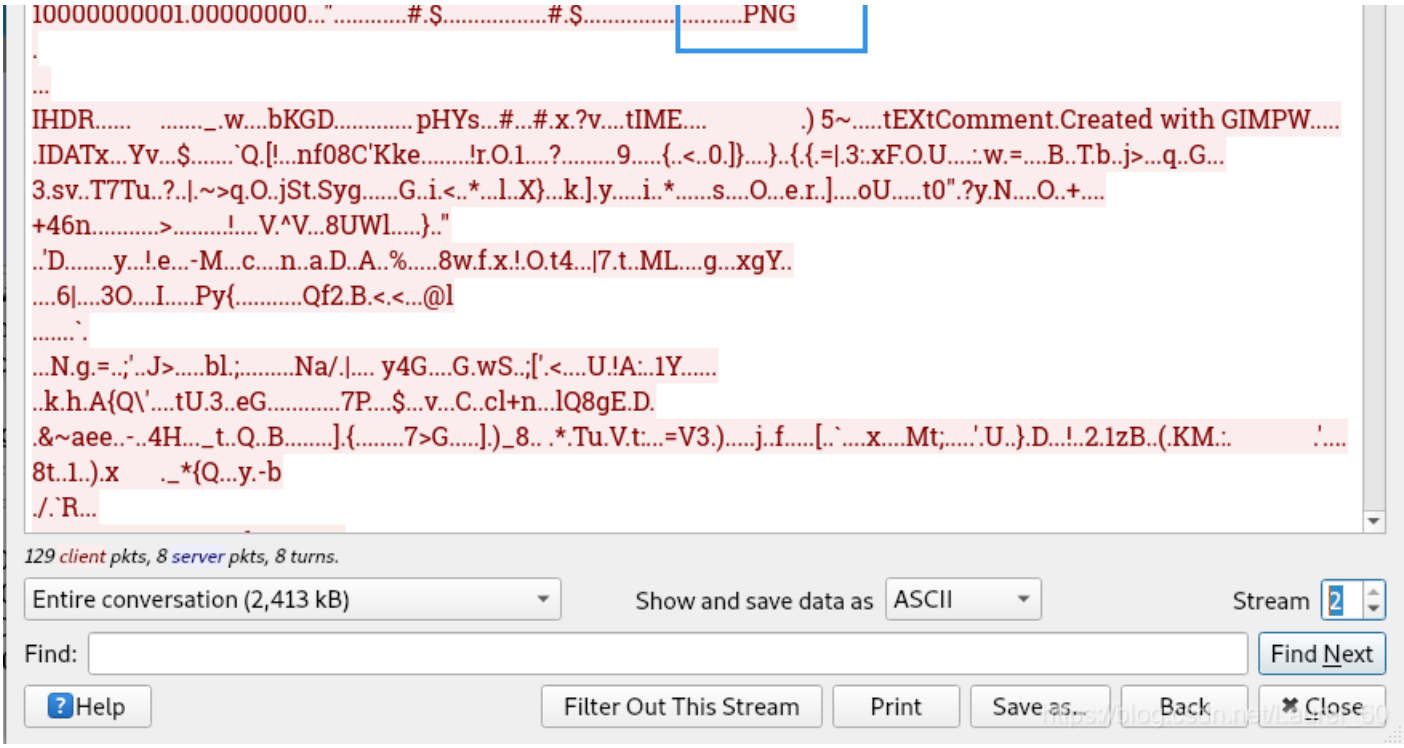


然后会看到有一个flag.png- -

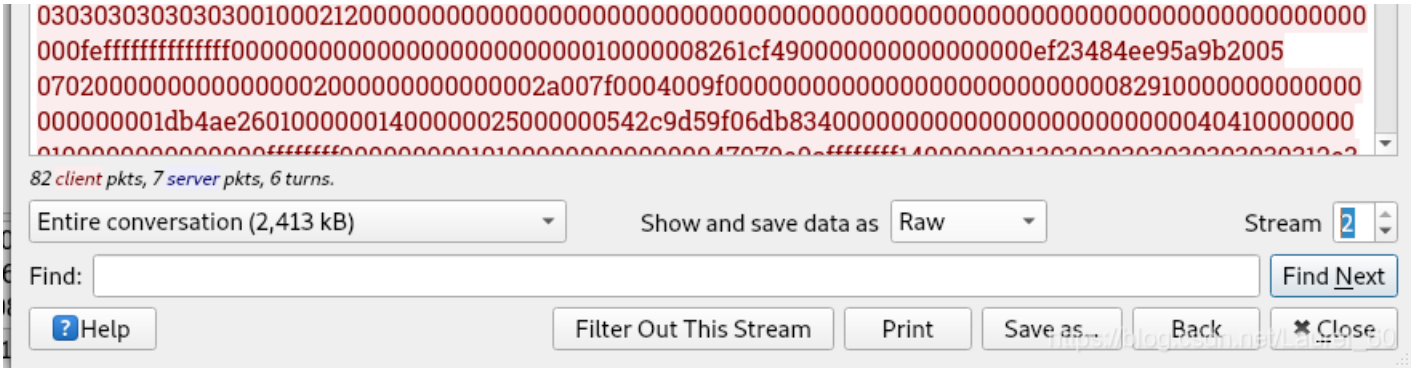


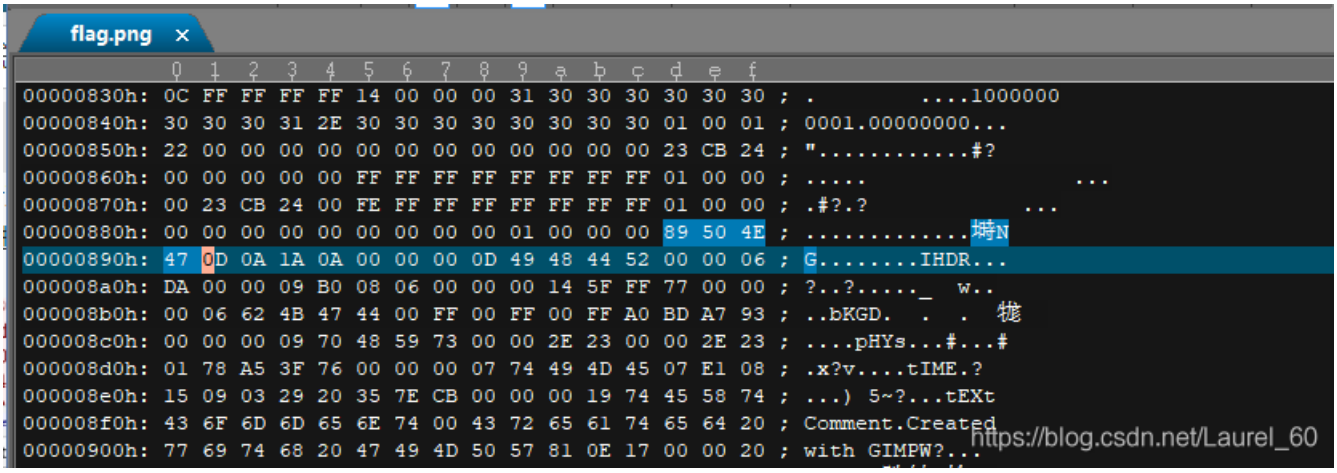
追踪一下TCP流，只有3个流，0号和1号流都没啥东西，3号流很长很长，而且有很明显的PNG图片标志- -





直接把这个流的数据导出来，删掉前面的就行了。建议用原始数据导出来，然后用十六进制编辑器打开，这样会比较方便。找到`89594E47`，把前面的删掉，导出来就是flag了--





需要注意的就是，Wireshark在转换成原始数据的时候可能会有点慢，所以要注意一下左下角的进度，不然如果在没转完的情况下就保存的话就只能保存一部分鸭?

№.3 NTFS交换数据流隐写

NTFS数据流指的是微软Windows NT内核的系列，是NTFS磁盘格式的一个特性。在NTFS文件系统中存在着NTFS交换数据流（Alternate Data Streams，简称ADS），每一个文件，都有着主文件流和非主文件流，主文件流能够直接看到；而非主文件流寄宿于主文件流中，无法直接读取，这个非主文件流就是NTFS交换数据流。

另外，安利一个贼好用的数据流分析软件：NtfsStreamsEditor
普通网站下载的有点bug，导出不了，还是师傅的比较好用：

> http://blog.sina.com.cn/s/blog_5f4838d10100dedc.html

直接管理员权限打开，就能分析出文件有没有包含数据流。比如下面这道题。

日常做题：

拿到一个压缩文件Misc150.rar，大小为427KB，普通解压之后，只得到一个Misc150.txt文件，大小为1KB，所以肯定有猫腻。而该txt文件内容是--
Flag.zip behind me.
我们用winrar在cmd中解压，我刚开始使用的命令如下--

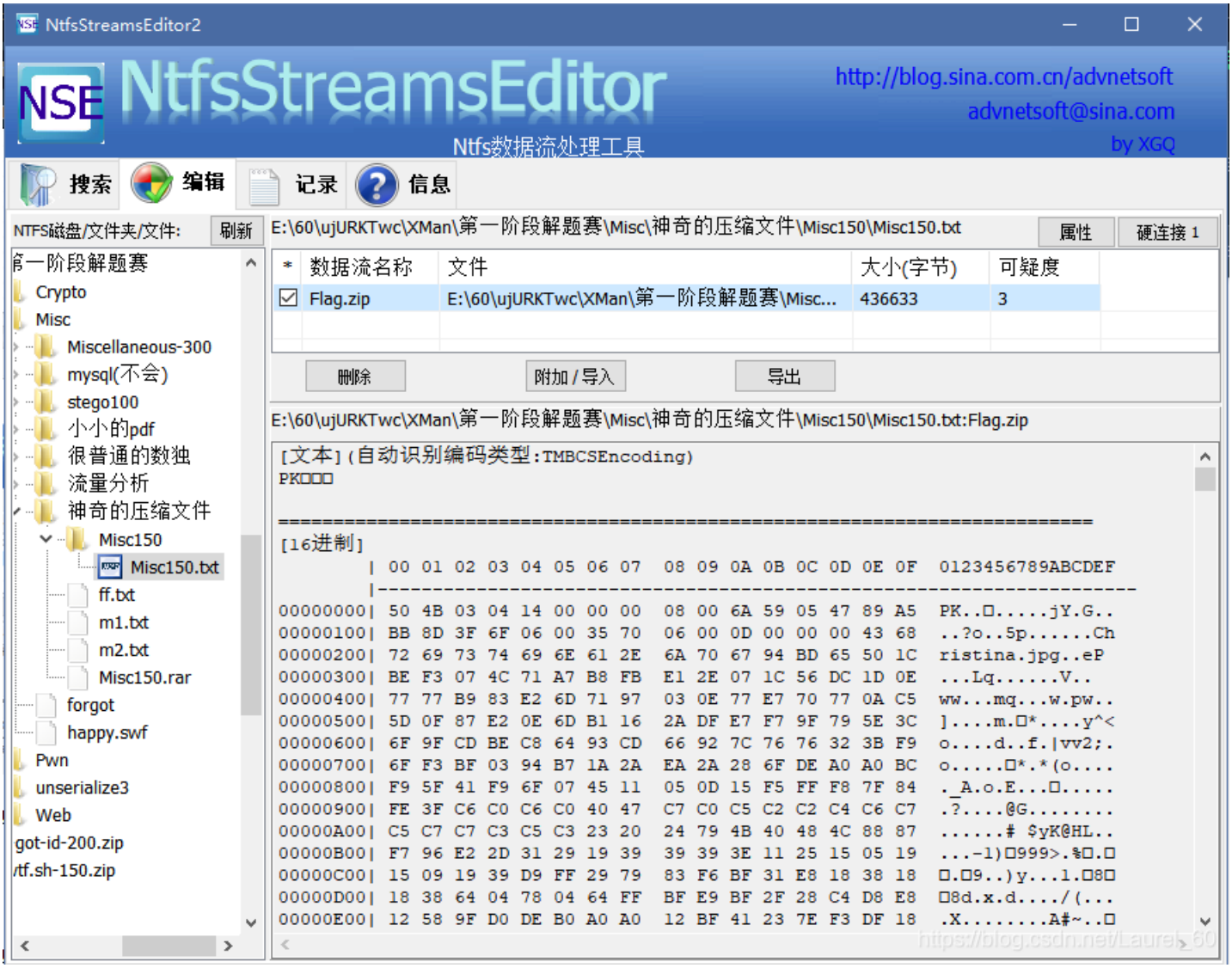
```
"C:\Program Files\WinRAR\winRAR.exe" C:\Users\**\Documents\Misc150\Misc150.txt:Flag.zip
```

但是enter之后发现报错，找不到压缩文件，就很奇怪。试过很多次觉得有可能是存储的路径含有中文的缘故，就换了一下路径，如下--

```
"C:\Program Files\WinRAR\winRAR.exe" D:\Misc150\Misc150.txt:Flag.zip
```

就可以看到隐藏的文件。

或者直接用NtfsStreamsEditor看看，导出就完事了 - -



菜鸡瑟瑟发抖