

MISC系列——图片隐写初阶套路（持续填坑中）

原创

SchellT  于 2019-05-31 21:28:46 发布  1528  收藏 10

分类专栏: [MISC系列](#) 文章标签: [misc ctf](#) [信息安全](#) [网络安全](#)

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/qq_43470389/article/details/90722379

版权



[MISC系列](#) 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

方法汇总

- [winhex的使用](#)
- [查看图片属性](#)
- [图片高宽度修改引起的隐写](#)
- [LSB隐写](#)
- [帧数查看](#)
- [复合文件隐写](#)

注意都是初阶套路, misc水很深, 能淹死人的那种。

有时需要多项结合, 望诸君充分利用, 实践出真知。

winhex的使用

winhex 是一个专门用来对付各种日常紧急情况的工具。它可以用来检查和修复各种文件、恢复删除文件、硬盘损坏造成的数据丢失等。同时它还可以让你看到其他程序隐藏起来的文件和数据。总体来说是一款非常不错的 16 进制编辑器。

现在就来基本介绍一下其在ctf中的使用：

如这张图片，下载后用winhex打开，就能看到如下标蓝的字符串：

A2 8A 00 FF 26 23 31 30 37 3B 26 23 31 30 31 3B	ç! ýke
26 23 31 32 31 3B 26 23 31 32 33 3B 26 23 31 32	y{
31 3B 26 23 31 31 31 3B 26 23 31 31 37 3B 26 23	1;ou&#
33 32 3B 26 23 39 37 3B 26 23 31 31 34 3B 26 23	32;ar&#
31 30 31 3B 26 23 33 32 3B 26 23 31 31 34 3B 26	101; r&
23 31 30 35 3B 26 23 31 30 33 3B 26 23 31 30 34	#105;gh
3B 26 23 31 31 36 3B 26 23 31 32 35 3B D9 D9	;t}Û

unicode解码即可。

key{you are right}儻

key{you are right}

https://blog.csdn.net/qq_43470389

很多类低阶图片隐写都可以用winhex查看得出。
网上资源很多，要是实在没找着还可以用010editor代替。

查看图片属性

这个没啥好讲的。

一张图洞破天机。



照相机制造商
照相机型号 73646E6973635F32303138
光圈值
曝光时间
ISO 速度
曝光补偿
焦距
最大光圈
测光模式
目标距离
闪光灯模式
闪光灯能量
35mm 焦距
高级照片
镜头制造商
镜头型号
闪光灯制造商

[删除属性和个人信息](#)

<https://blog.csdn.net/qj1403470389>

虽然这个没啥有用的，一般套路就是base64什么加密也没啥意思.....

图片高宽度修改引起的隐写

[题目下载](#)

下载后解压缩为一张图片，继续使用前面提到的工具winhex打开。

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI	ASCII
0	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	!PNG	IHDR
16	00	00	01	F4	00	00	01	A4	08	06	00	00	00	CB	D6	DF	ô	x
32	8A	00	00	00	09	70	48	59	73	00	00	12	74	00	00	12	!	pHYs
48	74	01	DE	66	1F	78	00	00	0A	4D	69	43	43	50	50	68	t	þf x
64	6F	74	6F	73	68	6F	70	20	49	43	43	20	70	72	6F	66	o	toshop ICC prof

在继续操作前，先来了解下PNG的文件格式。

上图的89 50 4E 47是PNG文件的文件头。

而从IHDR开始就是文件头数据块。

此后的八个字节就是图片宽高的值。

详细参考下图：

IHDR

文件头数据块IHDR(header chunk)：它包含有PNG文件中存储的图像数据的基本信息，并要作为第一个数据块出现在PNG数据流中，而且一个PNG数据流中只能有一个文件头数据块。

文件头数据块由13字节组成，它的结构如下表所示

又件头数据块由13字节组成，它的格式如下表所示。

域的名称	字节数	说明
Width	4 bytes	图像宽度，以像素为单位
Height	4 bytes	图像高度，以像素为单位
Bit depth	1 byte	图像深度： 索引彩色图像：1, 2, 4或8 灰度图像：1, 2, 4, 8或16 真彩色图像：8或16
ColorType	1 byte	颜色类型： 0: 灰度图像, 1, 2, 4, 8或16 2: 真彩色图像, 8或16 3: 索引彩色图像, 1, 2, 4或8 4: 带α通道数据的灰度图像, 8或16 6: 带α通道数据的真彩色图像, 8或16
Compression method	1 byte	压缩方法(LZ77派生算法)
Filter method	1 byte	滤波器方法
Interlace method	1 byte	隔行扫描方法： 0: 非隔行扫描 1: Adam7(由Adam M. Costello开发的7遍隔行扫描方法) http://log.csdn.net/qq_43470389

再继续操作，把高宽值改一样，这里的原理：

只要宽高大于原图的宽高就可以把全图显示出来。

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI	ASCII
0	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	!PNG	IHDR
16	00	00	<u>01 F4</u>	00	00	<u>01 F4</u>	08	06	00	00	00	CB	D6	DF			ó	ó
32	8A	00	00	00	09	70	48	59	73	00	00	12	74	00	00	12	!pHYs	t

注意这里是把高宽改大，一些二五仔却总是喜欢把它改小解不出来问为什么.....
最后打开图片就可以看到gf了。



BUGKU{a1e5aSA}

LSB隐写

选用真——脑洞比赛iscc的题来讲解一下，最后flag真的弄得我心力交瘁.....





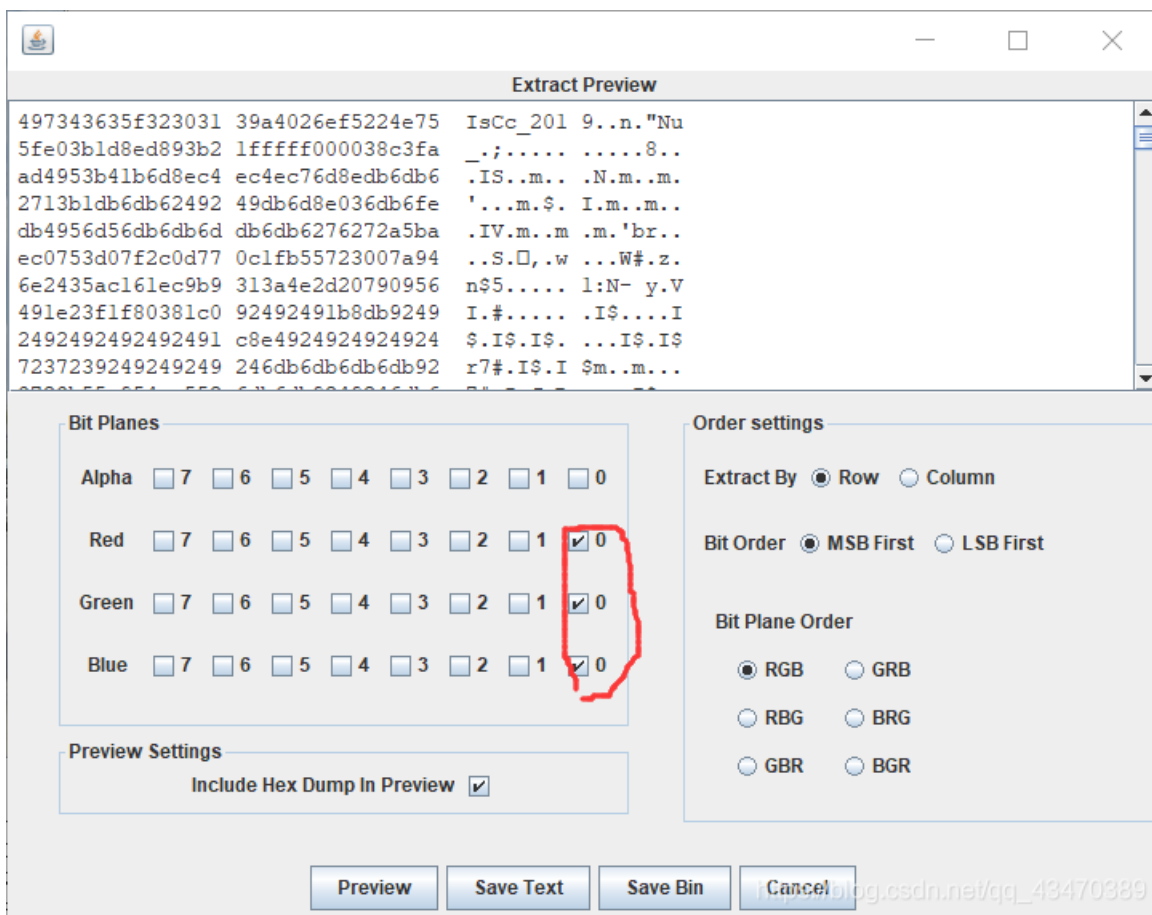
上述套路验过了都不行，就想了是不是lsb隐写，这里要用到一个工具StegSolve。

注意，这个软件要配置JAVA环境。

[戳我](#)

下载好后打开这个图片。

Analyse->Data Extract



上述分别是红绿蓝通道，原理太复杂还要扯到RGB，ctf练习生不需要了解。

所以，Preview就可以看到上面那个iscc之类的了。

到这里我还是要吐槽一下，你个flag格式都不给，弄半天恶心心???



帧数查看

题目下载

下载后发现是个扭曲的二维码，得想办法暂停一下。

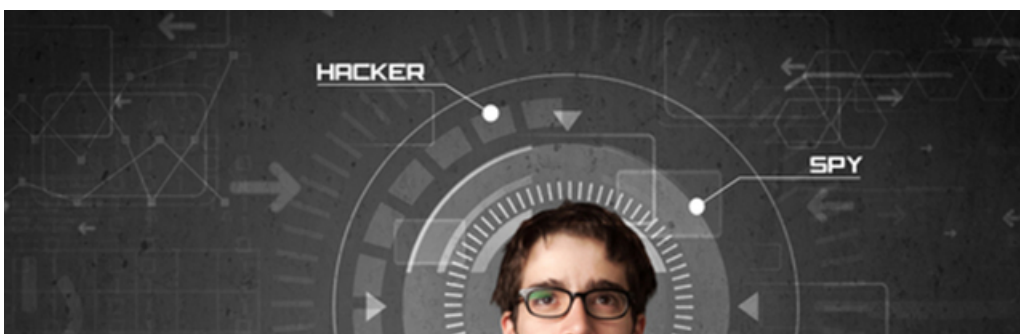
继续用到上面的工具StegSlove打开gif。

Analyse->Frame Browser



打开就用下面的箭头一个一个点就是了，挨个用手机扫拼起来就是flag。

复合文件隐写





如上图，上述套路都莫得用。

不用想了，binwalk。

kali虚拟机操作。

```
root@kali: ~/question
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~# cd question
root@kali:~/question# binwalk 2.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, EXIF standard
12	0xC	TIFF image data, big-endian, offset of first image directory: 8
158792	0x26C48	JPEG image data, JFIF standard 1.02
158822	0x26C66	TIFF image data, big-endian, offset of first image directory: 8
159124	0x26D94	JPEG image data, JFIF standard 1.02
162196	0x27994	JPEG image data, JFIF standard 1.02
168370	0x291B2	Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"

https://blog.csdn.net/qq_43470389

如上，发现捆绑了好几个文件，从158792块开始就是另一jpg文件了。

所以使用dd。

```
root@kali: ~/question
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~# cd question
root@kali:~/question# binwalk 2.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, EXIF standard
12	0xC	TIFF image data, big-endian, offset of first image directory: 8
158792	0x26C48	JPEG image data, JFIF standard 1.02
158822	0x26C66	TIFF image data, big-endian, offset of first image directory: 8
159124	0x26D94	JPEG image data, JFIF standard 1.02
162196	0x27994	JPEG image data, JFIF standard 1.02
168370	0x291B2	Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"

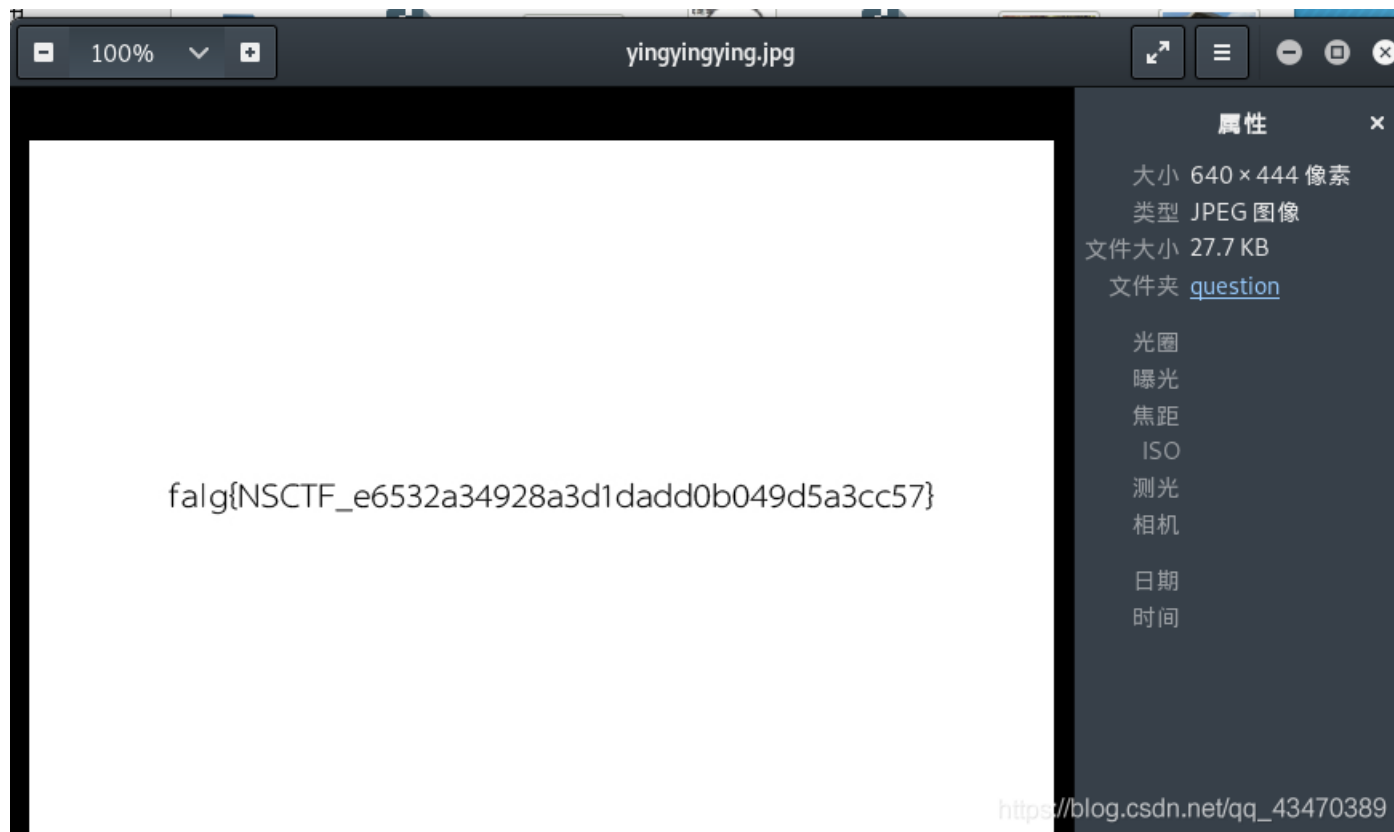
```
root@kali:~/question# dd if=2.jpg of=yinyingying.jpg skip=158792 bs=1
记录了 27689+0 的读入
记录了 27689+0 的写出
27689 bytes (28 kB, 27 KiB) copied, 0.0696103 s, 398 kB/s
root@kali:~/question#
```

https://blog.csdn.net/qq_43470389

这里解释一下：

if为上述测试的文件名，of为输出的文件，skip为跳过的数据块，bs为跳过一个的意思。

这样就分离出了嘤嘤嘤。



许多题目都是复合文件，binwalk还大有用处，有道是 kali用得好，局子.....手动滑稽。