图片隐写术总结

转载



【<u>网络安全/渗透测试】 专栏收录该内容</u> 3 篇文章 0 订阅

订阅专栏

0x00 前言

之前还没有见到drops上有关于隐写术的总结,我之前对于隐写术比较有兴趣,感觉隐写术比较的波玩。所以就打算总结总结一些隐写术方面的东西。写的时候,可能会有错误的地方,请不吝赐救,谢谢。

本篇章中用到的隐写术的图片,都打包在了http://pan.baidu.com/s/1mg1Khw0,想去自己尝试一遍的话可以去下载。

最开始接触到隐写术,是看到一种叫做图种的东西,当时不懂,只说要另存为zip,然后解压出来就可以了,当时觉得特别种奇,就像发现了新大陆,然后就尝试了一下,发现可以用另存为zip的方式,用7z或者是winzi p等工具打开,然后就可以看到福利了。



图1.png

后来才懂得了,先制作一个1.zip,把想要随意的东西放进去,再需要一张pg图片2.jpg,然后就可以执行一个命令 copy /b 2.jpg+1.zip output.jpg。就可以得到一张图种,这是利用了 copy命令,将两个文件已二进制 方式组接起来,生成output.jpg的新文件。而在jpg中,是有结束符的,16进制是FFD9,利用 winhex可以看到正常的 jpg结尾都是FFD9的,图片查看器会忽视 jpg结束符之后的内容,所以我们 动加的zip,自然也就不会影 响到图像的正常显示。



图2.png

这种类型的隐写也是比较容易被发现的,如果发现是jpg图片的话,观察文件结束符之后的内容,查看是否附加的内容,正常图片都会是FF D9结尾的。还有一种方式来发现就是利用binwalk这个工具,在kali下自带的 个命令行工具。

root@kali:~	/Desktop/demo# b	inwalk 9b48d91f3a292df51ad	5ffbdbc315c6035a873ae.jpg
DECIMAL	HEX	DESCRIPTION	
151728	0×250B0	End of Zip archive	360安全 播报(bobao.360.c n)

图片3.png

利用binwalk可以自动化的分析图片中附加的其他的文件,其原理就是检索匹配文件头,常用的一些文件头都可以被发现,然后利用偏移可以配合winhex或者是dd分割出隐藏的给分。

0x01 修改数据

上面说到的隐藏方式,是利用了增加数据的方式,把数据直接增加在了jpg后面。还有另一类隐藏的方法,就是利用了修改数据的方式来隐藏自己传递的信息。

一种常见的方式是利用LSB来进行隐写,LSB也就是最低有效位(Least Significant Bit)。原理就是图片中的像数一般是由三种颜色组成,即三原色,由这三种原色可以组成其他各种颜色,例如在PNG图片的储存中,每 个颜色会有 8bit,LSB隐写就是修改了像数中的最低的1bit,在人眼看来是看不出来区别的,也把信息隐藏起来了。譬如我们说明"(A'隐藏进来的话,如下图,就可以把A转成16进制的0x61再转成二进制的01100001,再 修改为红色通道的最低位为这些二进制串。



图4.png

如果是要寻找这种LSB隐藏痕迹的话,有一个工具是个神器,可以来辅助我们进行分析。Stegsolve这个软件的下载地址是

http://www.caesum.com/handbook/Stegsolve.jar

打开之后,使用Stegsolve——Analyse——Frame Browsen这个可以浏览三个颜色通道中的每一位,可以在红色通道的最低位,发现一个二维码,然后可以扫描得到结果。



图6.png

再解一下qrcode,用在线的就可以http://tool.chinaz.com/qrcode/,得到了flag(AppLeU0),如果是隐写的使用了ascii的话,可以使用Stegsolve——Analyse——Data Extract来查看ascii码。

在这个过程中,我们要主意到,隐写的载体是PNG的格式,如果是像之前的jpg图片的话就是不行的,原因是jpg图片对像数进行了有损的压缩,你修改的信息可能会被压缩的过程破坏。而PNG图片虽然也有压缩,但 却是无损的压缩,这样子可以保持你修改的信息得到正确的表达,不至于丢失。BMP的图片也是一样的,是没有经过压缩的,可以发现BMP图片是特别的大的,因为BMP把所有的像数都按原样储存,没有压缩的过程。

0x02 隐写与加密

我们先要区分一个概念,隐写术和咖啡密的区别。其实说起来很简单,加解密的话,就是会出现一些神秘的,可疑的字符串或者是数据之类的。而隐写术的 话,就是信息明明就在你的面前,你起对他视而不见。隐写术 在CTF中出现时,常常会和加解密结合起来一起出现,或者是一些编码方式一起出现,以提高题目的 难度。

用一个ttf的题目作为例子吧,iscc2014中有一个题目,给了一个名为此为gif图片:gif的文件,打开发现了报错。有的时候,会需要我们去修复图片,这对我们对于图片的文件结构要有了解。找到gif的文件格式,然后 对照 这个破损的文件。Gif的图片格式文档可以查看这个链接,http://dev.gameres.com/Program/Visual/Other /GIFDoc.htm

Offset	0	1	- 2	- 3	- 4	- 5	6	- 7	8	- 9	A	В	C	D	E	F		٨
00000000	39	61	A2	06	6B	04	F7	FF	00	20	20	20	02	02	02	23	9a? k. ?#	_
00000010	23	23	04	04	04	2B	2B	2B	21	21	21	06	06	06	33	33	##+++ <mark>!!!.</mark> 33	
00000020	33	05	05	05	FE	FE	FE	28	28	28	27	27	27	2D	2D	2D	3 ? (('''	
00000030	3C	3C	30	51	51	51	30	2D	2E	CD	CD	CD	D3	D3	D3	4ô	// / / 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、	rg

图片8.png

用winhex打开,我们会发现他和普通的GIF图片不一样,头部缺少了东西,在对北一些文档,会发现是少了GIF8。



图片9.png

我们手动修复一下,增加GIF8。

Offset	0	1	2	3	4	5	6	7	8	9	A	В	С	D	E	F	^
00000000	47	49	46	38	39	61	A2	06	6B	04	F7	FF	00	20	20	20	GIF89a <mark>?kk.</mark> ? .
00000010	02	02	02	23	23	23	04	04	04	2B	2B	2B	21	21	21	06	
00000020	06	06	33	33	33	05	05	05	FE	FE	FE	28	28	28	27	27	

图片10.png

然后浏览图片后会发现,有个PASSWORD一闪而过,gi环岛的图片最大的区别就是gif是动态图,它是可以由多帧组成的可以顺穷播放的,有的感就是把播放的时间弄得特别慢,几乎就不会动的,所以我们可以用工具一帧一帧的观察图片。Stegsolve就带有这种功能。

Stegsolve—Analyse—Frame Brower就可以看到是有8帧的图片,有点重叠不太好观察,也可以用Namo_GIF_gn这个工具。得到了PASSWORD is Y2F0Y2hfdGhlX2R5bmFtaWNfZmxhZ19pc19xdW10ZV9zaW 1wbGU=。很明显,这个时候PASSWORD是经过的编码的,我们可以看到字符范围是0-9a-Z结尾还有=,所以判断是base64编码,解码得到了 catch_the_dynamic_flag_is_qurnte_simple。这个就是和编码方式结合,传 递一些可疑的激励,隐写术常常会与加解密或编码结合在一起,对一些常见的编码和加密方法也更了解,得到密文的字符范围和长度能发现这是什么加密或者是编码。

0x03 载体

数据在隐藏的时候,我们常常是需要先分析是数据隐藏在哪里,也就是他在利用是什么做载体,之后才可以进一步的分析是加密或编码的。这也就是说我们要对一个图片的格式要有了解,才能知道哪些地方是可疑的,哪些是可以隐藏起信息的,会有冗余的成分在。举个例子吧,比如给了一个jpg的图片。除了我们之前说到的隐藏在结束符之后的信息,jpg图片还可以把信息隐藏的exif的信息是jpg的头部插入了数码照片的信息,比如是用什么相机拍摄的。这些信息我们也是可以控制的,用查看属性的方式可以修改一部分的信息,还可以用exif编唱器来进行编辑。Power_exif这个可以用来编辑。

🔹 exif.jpg 属性	10.00	×
常规 安全	详细信息以前的版本	
属性	值	^
说明 ——		-
标题		E
主题		
分级		
标记		
會 汪	flag(AppLeUU)	
来源		- 11
作者		
- 拍摄口期 程序复数		
· 拉所有称 		
版权		
图像		-
图像 10		
尺寸	630 x 354	
宽度	630 像素	-
除属性和个		
	确定 ₆₀ 。即消,成	用 (A)

图片11.png

可以看到flag(AppLeU0),就是需要了解除藏信息的地方,隐写术有的时候难,就是难在了一张图片有太多的地方可以隐藏信息了,有的时候根本连隐藏的载体都找不到,在你的眼里他就是一张正常的图片。 0x04 编程辅助 有一些情况下,我们也是没有现成的工具来完成的,可以自己写一些简单的程序来辅助我们进行分析,或者是加解密。比如sctf的misc400的题目,就需要用到一些简单的编程。题目给出了一个png图片,需要我们找到 有SCTF(物志的flag。

这个题需要我们对于png图片的格式有一些了解,先用stegsolve查看一下,其他的LSB之类的并没有发现什么问题,然后看了一下结构发现,有一些异常的IDAT块。IDAT是png图片中储存图像像数数据的块。Png图片格式的扩展阅读可以看看这篇

http://www.cnblogs.com/fengyv/archive/2006/04/30/2423964.html

有详细的介绍。

🌋 File Format Analysis	
Data longui - +Juzz bytos	-
CRC = 68958fcd	
Image data, compressed	
Chunk:	
Critical - necessary for display of image MUST BE recognized to	
proceed	
Public	
Unsafe to copy unless known to software	
Hex:	
49444154	
Ascii:	
IDAT	
Data length = 138 bytes	
CRC = d9cfa5a8	
Image data, compressed	
Chunk:	-
05	
drops.wooyun.	org

图片12.png

可以用pngcheck来辅助我们观察,可以看得更加清晰。pngcheck.exe -v sctf.png

c hun k	IDAT	at	offset	0×130008,	length	65524
c hun k	IDAT	at	offset	0×140008,	length	65524
c hun k	IDAT	at	offset	0×150008,	length	45027
c hun k	IDAT	at	offset	Øx15aff?,	densities	129 hohao 260 co \
c hun k	I END	at	offset	0x15b08d,	length	mg (bobao.soo.cn)

图片13.png

可以看到,正常的块的length是在65524的时候就满了,而倒数第二个IDAT块长度是45027,最后一个长度是138,很明显最后一个IDAT块是有问题的,因为他本来应该并入到倒数第二个未满的块里。

																	The second se
1421296	95	8F	CD	00	00	00	8A	49	44	41	54	78	90	5D	91	01	??1?IDATx?]?.?
1421312	12	80	40	08	02	BF	04	FF	FF	5C	75	29	4B	55	37	73	. 60?. u)KU7s
1421328	8A	21	A2	7D	1E	49	CF	D1	7D	B3	93	7A	92	E7	E6	03	7% .I涎}?摸掔?.
421344	88	0A	6D	48	51	00	90	1F	BO	41	01	53	35	0D	E8	31	行mHQ殤.S5.?1.
421360	12	EA	2D	51	C5	4C	E2	E5	85	B1	5A	2F	C7	8E	88	72	.?-Q穩忮叡Z/荚坮
421376	F5	10	6F	C1	88	18	82	F9	3D	37	2D	EF	78	E6	65	BO	??o??₹債=7-?x?e?
421392	C3	60	52	96	22	A0	A4	55	88	13	88	33	A1	70	A2	07	胠R?"? ?.?3 ?.
421408	1D	DC	D1	82	19	DB	80	0D	46	5D	8B	69	89	71	96	45	. ?褌. ??.F]姐塹朎
421424	ED	90	11	C3	6A	E3	AB	DA	EF	CF	CO	AC	FO	23	E7	7C	項.?j? 锵垃??鐊?
421440	17	C7	89	76	67	D9	CF	A5	A8	00	00	00	00	49	45	4E	.?壞g?膝 IEN
421456	44	AE	42	60	82												C03°.
																	drops.wooy

图片14.png

我们用winhex把这一部分异常的IDAT块给扣出来。然后就是要研究研究这个块是什么情况,发现了载体之后就是要想办法找出他的规律。观察那一部分的数据,可以看到是16进制的78 9C开头的,百度一下分析是zli b压缩的标志。在png的百度百科里也可以查到PNG的IDAT是使用从LZ77派生的无损数据压缩算法,可以用 zlib解压。那么就尝试用zlib来解一下这段数据。Zlib的扩展阅读http://zlib.net/

我们使用python来编程,先把那段数据处理一下,保存成16进制的。

80	28	4A	80	00	00	00	8A	49	44	41	54	78	9C	5D	91	01	(J€Š <mark>IDAT</mark> x?]?.
96	12	80	40	08	02	BF	04	FF	FF	5C	75	29	4B	55	37	73	. €0?. u)KU7s
112	8A	21	A2	7D	1E	49	CF	D1	7D	B3	93	7A	92	E7	E6	03	? . I涎}?擤掔?
128	88	0A	6D	48	51	00	90	1F	BO	41	01	53	35	0D	E8	31	? mHQ癆.S5.?
144	12	EA	2D	51	C5	4C	E2	E5	85	B1	5A	2F	C7	8E	88	72	.?-Q臠忮叡Z/美坮
160	F5	1C	6F	C1	88	18	82	F9	3D	37	2D	EF	78	E6	65	BO	? o?? 債=7-?x?e?
176	C3	60	52	96	22	A0	A4	55	88	13	88	33	A1	70	A2	07	胠R?"? ? ? ?
192	1D	DC	D1	82	19	DB	80	0D	46	5D	8B	69	89	71	96	45	.?褌.??F]媔%a脫
208	ED	90	11	C3	6A	E3	AB	DA	EF	CF	CO	AC	FO	23	E7	7C	頊.?j?«陲侠 #?
224	17	C7	89	76	67	D9	CF	A5	A8	00	00	00	00	49	45	4E	.?塿g?膝 IEN
240	44	AE	42	60	82												DYB drops.wooyun.org

图片15.png

得到16进制的以方便python处理,前面的4字节是长度然后是标志位IDAT然后开始是数据,直到 D9 CF A5 A8是crc32校验位。 所以实际的数据是:

788<50910112804008028P04FFFSC752948553773821827024957017083937A2275603880 A6048510901Fm941013335008311282A2015445285385135827585872F5145472451458188279 3037202F78645380345459425858138833A170A2071D0021821508500445588598919 458209411545381ABAREFCF0427021F70217059657

然后用python来写zlib解压

	#! /usr/bin/env python
	import zlib
1	import binascii
2 3 4	IDAT = "75995091011280400802BF04FFFF5C75294B5537738A21A27D1E49CFD17DB3937A92E7E603880A6D485100901FB0410153350DE83112EA2D51C54CE2E585B15A2FC78E8872F51C6FC1881882F93D372DEF .decode('hex')
6	#print IDAT
7 8	result = binascii.hexlify(zlib.decompress(IDAT))
	print result
	<pre>#print result.decode('hex')</pre>

发现解出来了一些3031的字符串,30和31是hex的0和1的编码,再解一次hex得到一串625长度的01字符串。

11111110001000011011111111000001011100101
111010010000000010111011011010110100101110110000
111010101010101111111000000001011101110000
010100100001110000000000101000000010010
10111110001100101000110011100001010100011010
01110110010000111001110010000101111111010
0001101011011000001000011001100011110101
1101001110010111010010011101101000110000
111011011

得到的01 串的长度是625,除以8 除以7 都无法整除,也就是说没法直接转换成ascii码。



图片16.png

然后发现625 = 25*25,刚好是个正方形的形状,那么尝试一下把这些01组成一个正方形看着是什么,可以用python的PIL编程可以很方便的画图,在kali自带就可以有,win的环境需要安装PIL的第三方库。

	#!/usr/bin/env python
	import Image
	MAX = 25
	pic = Image.new("RGB" ;(MAX MAX))
	str = "1111110001000011011111110000010111001011010
	i = 0
1 2	for y in range (0



SCTF{(121.518549,25.040854)},成功得到了flag。

0x05 双图

发现是一个二维码可以编码来画出 0代表了是白色而1代表了黑色,然后可能会需要旋转来调整一下,才能扫描出来。处理一下得到了一个二维码。然后扫描得到了flag。

在有的情况下,是设法用现成的工具来处理的,所以就要我们用编程来设法解决。Python的PIL是个好东西。批量处理图片的时候可能会需要它。

个时候就要用到一些对此的技巧来查找了。比如ISG2014的misc200就是用到的这种给出了两张图的。有的情况下,第二张图是需要你自己去找到的。 我们来看isg2014-misc200的题,题目给了一张png图片,png的图片,就怕里面插个什么rar之类的, 所以先用linux下的binwalk命令跑一跑。



操作的类型: 🔘 生成 💿 解码

二维码图片: 选择图片 解码数据为: SCTF{(121.5)3549,25.0408549; arops.wooyun.org

图片18.png

还有一种情况是比较特殊的,有的时候会给出两张图片,或者是需要你去寻找原来的图片来进行对比寻找隐藏的信息。这个一般是因为一张图片给出来的隐藏信息太过于隐蔽,无去找不到具体的位置,具体的信息。这

PNG image, 1440 × 360 条前,你们的人们的人们的人们的

HEX

图片19.png

跑一跑,发现了有两个PNG图片,binwalk会给出偏移,确定了偏移是0x1D55DC之后,用winhex把图片扣出来,保存成2.png。原来的图final.png删除后面的一部分,保存成1.png。肉眼查看了一下,发现两张图 片没有太大的区别,我们用软件来帮助我们区分他。

001D55D0	00	00	00	00	49	45	4E	44	AE	42	60	82	89	50	4E	47	IEND@B`IIPNG
001D55E0	OD	0A	1A	0A	00	00	00	OD	49	48	44	52	00	00	05	AO	IHDR
001D55F0	00	00	03	84	08	02	00	00	00	D8	2F	01	85	00	01	00	1 0/1
001D5600	00	49	44	41	54	78	9C	BC	FD	57	B3	6C	4B	92	1E	88	IDATx1%ýW°1K″
001D5610	B9	88	88	25	52	6D	71	F4	B9	AA	EA	56	75	55	0B	34	¹∎%Rmgô¹≇éVuU 4
001D5620	80	86	26	7B	0C	63	ΕO	DO	86	0F	63	9C	67	3E	FO	F7	II&{ càĐI cIg>ð÷
001D5630	F1	85	C6	17	92	83	01	8D	30	62	80	69	88	6E	4C	8B	ñ Æ 1 OblinL
001D5640	EA	AE	ΕA	52	57	1F	BD	65	8A	A5	22	C2	DD	79	36	32	éBolai selstava 2
00105650	73	E7	D9	67	9F	DB	55	40	01	61	D7	CE	CD	BD	73	ED	scUg UUP ax 18s1

图片20.png

用linux下的命令可以进行对比,生成一个有差异的图片diff.png。compare 1.png 2.png diff.png 观察一下发现了左下角有异常, png图片像数保存是从左到右,从下往上排列的。



图片21.png

发现了左下的第二条像素有异常,对比一下1.png 2.png发现了2.png有问题那么我们可以用神器stegsolve来辅助,stegsolve——Analyse——Image Combiner对比两个文件。查看Sub或Xor,可以发现左下角,第二条像数条是有异常的,有红色的出现。



图片22.png

把1.png和2.png进行一下sub方法把结果保存成solved.bmp。

然后把2.png保存成2.bmp 24位位图的格式,这个是因为png图片经过了压缩,不好直接机步带,而bmp图片是没有压缩的,直接保存各个像数点的数据。

这个题还有一个坑点就是偏移的问题 png图片的扫描是从左向右,从下往上来的。而坑的是这个图的信息隐藏并没有在一开头的缘数,而是是第二行缘数,所以就需要利用bmp的优势,储存无压缩,方便寻找到偏移 ,从而找到信息隐藏的地方。利用winhex打开,黑色的像数的在bmp中的hex的00保存的,那么我们就寻找不是00的地方。在偏移0x1110的地方可以发现

00001110	00	00	00	00	00	00	00	00	34	00	00	33	00	00	34	00	4 3 4
00001120	00	33	00	00	32	00	00	33	00	00	33	00	00	31	00	00	3 2 3 3 1
00001130	32	00	00	31	00	00	32	00	00	30	00	00	31	00	00	31	2 1 2 0 1 1
00001140	00	00	30	00	00	30	00	00	30	00	00	2E	00	00	2F	00	000./
00001150	00	2F	00	00	2E	00	00	2C	00	00	2C	00	00	2C	00	00	1
00001160	2E	00	00	2D	00	00	2D	00	00	2D	00	00	2D	00	00	2E	1. T. T. T. T. J.
00001170	00	00	2D	00	00	2D	00	00	2F	00	00	2E	00	00	2F	00	/ . /
00001180	00	2F	00	00	2F	00	00	2F	00	00	30	00	00	2F	00	00	///0/
00001190	2E	00	00	2E	00	00	2E	00	00	2E	00	00	2F	00	00	2E	/ .
000011A0	00	00	30	00	00	30	00	00	31	00	00	30	00	00	30	00	0 0 1 0 0
000011B0	00	30	00	00	31	00	00	31	00	00	30	00	00	30	00	00	0 1 1 0 0
000011C0	31	00	00	30	00	00	30	00	00	2F	00	00	2F	00	00	30	1 0 0 / / 0
000011D0	00	00	2F	00	00	2E	00	00	30	00	00	2F	00	00	2F	00	/ . 0 / /
000011E0	00	2F	00	00	2E	00	00	2F	00	00	2E	00	00	2F	00	00	1.1.1
000011F0	30	00	00	2F	00	00	30	00	00	2F	00	00	30	00	00	30	drops.wboyth.or
00001000	00	00	277	00	00	077	00	00	20	00	00	377	00	00	277	00	1 1 0 1 1

图片23.png

有不是00的字节,一开始还以为这些就是flag的信息了,后来才发现是因为两个图片sub影响到了效果,真正的信息是隐藏在2.png中的,所以打开由2.png转换的2.bmp来对,通过之前diff得到的偏移,寻找到0x1110的地方,直到0x1330结束,这是隐藏的信息。

00001220	30 00 00	20 00 00 20 00	00 2r 00 00 30 00 00 30	0 / / / 0 0
000012F0	00 00 30	00 00 30 00 00	30 00 00 2F 00 00 30 00	0 0 0 / 0
00001300	00 30 00	00 2F 00 00 30	00 00 30 00 00 30 00 00	0 / 0 0 0
00001310	31 00 00	30 00 00 30 00	00 30 00 00 30 00 00 31	1 0 0 0 0 1
00001320	00 00 31	00 00 30 00 00	31 00 00 30 00 00 30 00	1 0 1 0 0
00001330	00 30 00	00 30 00 00 30	00 00 31 00 00 30 20 08	0 0 0 1 0
00001340	00 00 00	00 00 00 00 00	00 00 00 00 00 00 00 00	drops.wooyun.org

图片24.png

2. bep	solved.	bap																
Off	set	0	1	2	3	4	5	6	7	8	9	A	В	С	D	Е	F	
0000	10E0	9A	CE	FC	9A	CE	FC	9A	CE	FC	9A	CE	FC	9A	CE	FC	9A	lîulîulîulîulîu
0000	10F0	CE	FC	9A	CE	FC	9A	CE	FC	9A	CE	FC	9A	CE	FC	9A	CE	Îu Îu Îu Îu Îu Îu
0000	1100	FC	9A	CE	FC	9A	CE	FC	9A	CE	FC	9A	CE	FC	9A	CE	FC	üllüllüllüllüllü
0000	1110	9A	CE	FC	9A	CE	FC	B6	61	00	B6	61	01	B6	61	00	B 5	lîülîü¶a ¶a ¶a μ
0000	1120	60	00	B 5	60	01	B5	60	00	B5	60	00	B4	5F	01	B4	5F	, h, h, h, , ⁻ , ⁻
0000	1130	00	B4	5F	01	B4	5F	00	B3	5E	01	B 3	5E	00	B3	5E	00	·_ ·_ an an an
0000	1140	B3	5E	01	B 3	5E	01	B1	5F	00	BO	5E	01	B2	5D	00	B2	3^ 3^ ±_ *^ 2] 2
0000	1150	5D	00	B2	5C	00	B3	5B	01	B3	5B	01	B3	5B	01	B3	5A] 2 0 0 0 0 0Z
0000	1160	00	B 3	5A	01	B 3	5A	01	B3	5A	01	B 3	5A	01	B3	5A	00	2 ° Z ° Z ° Z ° Z
0000	1170	B3	5A	01	B3	5A	01	B2	5B	00	B2	5B	01	B2	5B	00	B2	3Z 3Z 2[2[2[2
0000	1180	5B	00	B2	5B	00	B3	5C	01	B3	5C	00	B 3	50	111	50	50	[\$5,000 0N: 0N: 0N
0000	1190	00	B2	5C	00	B 3	5D	01	B3	5D	01	B3	5D	00	B 3	5D	01	2/ aldhawddyllarou

图片25.png

只保留0001,这个是因为RGB的关系,只隐藏在R通道里面了,其他通道都是图片的正常像数信息,过湖卓就可以了。

000100000100001000100000010100010000000
0010000010100010000000101010000010100010000
000101000101010001000000100000010001000101
00101010000000101000000000000010100000101
010100000000000000000000000000000000000

观察一下可以发现,而奇数位都是0,是多余的,把这些去除掉。直接把00替换成0,01替换成1就可以了。

0100100101010011010001110111101101000101
011011101000100010101100111001101000110110000
110000010010000111100101111101

得到了这个之后,可以发现他的长度是184,是8的倍数,把他转换成ascii码就可以了。可以使用JPK工具来进行转换,工具的下载的描接是www.wechall.net/applet/JPK_406jar。 对比2.bmp可以发现隐藏了一些00 01这些信息,把这一部分扣出来。

▲ JPocketKnife v4.06a	
File Edit Search General Ascii	Binary Number About /dev/null
History 1/1 Clear <	Binary Format OutRadix 16 BitsParBlock 3
	Binary Format the text and validate its input. You should call formative environmented

图片26.png

JPK-binary-binary to ascii

\scii	Binary	Number	A	bout	/de	ev/null
	Binary	Format				OutRad
_	Binary	To Ascii			_	ountau
	Eldet C	onvori cum	áhi	p's.w	زەە	- un:Wrg

就得到了flag, ISG{E4sY_StEg4n0gR4pHy}

这种就是利用的两张图片对比来寻找差异,从而找到信息隐藏的地方,这样子出题往往是因为一张图片能提供的信息太少。

0x06 后记

这个总结其实还是缺很多的,因为隐写术能写的东西太多了,比如jpg的冗余信息的压缩也可以隐藏进信息,还有其他的多媒体文件也可以进行隐写,例如音频文件,视频文件等等,有很多东西可以研究。一开始是觉得隐写术特别的有趣才接触图的,就像是在藏宝寻宝一样,特别近玩,希望你们也可以感受到这种快乐。欢迎大家和我交流,我的博客地址是http://appleu0.sinaapp.com/。

0x00 前言

在安全的大趋势下,信息安全越来越来受到国家和企业的重视,所以CTF比赛场次越来越多,而且比赛形式也不断的创新,题目也更加新颖有趣,对选手的综合信息安全能力有一个较好的考验,当然 更好的是能从比赛有所收获,不断学习和总结提升自己的信息安全能力与技术。转到CTF比赛上,通常在CTF比赛中常有与隐写术(Steganography)相关的题目出现,这里我们讨论总结图片隐藏文件分离 的方法,欢迎大家补充和交流:P

0x01分析

这里我们以图片为载体,给了这样的一样图片:



首先我们需要对图片进行分析,这里我们需要用到一个工具 <u>binwalk</u>,想要了解这个工具可以参考这篇 <u>Binwalk:后门(固件)分析利器</u>文章,以及 <u>kali官方对binwalk的概述和使用介绍</u>。 这里我们就是最简单的利用,在binwalk后直接提供固件文件路径和文件名即可:

binwalk carter.jpg

当我们使用这行命令后, binwalk就会自动分析这个jpg文件:

binwalk carter.jpg

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data. JFIF standard 1.01
382	0x17E	Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"
3192	0xC78	TIFF image data, big-endian, offset of first image directory: 8
140147	0x22373	JPEG image data, JFIF standard 1.01
140177	0x22391	TIFF image data, big-endian, offset of first image directory: 8

从上面的内容显然看得出来这个jpg文件还隐藏着另一个jpg文件,从140147块偏移开始就是另一张jpg。

0x02分离

在得到隐藏信息之后我们下一步就是把另一张jpg分离出,以下讨论几种方法:

(1)使用dd命令分离(linux/unix下)

我们可以使用dd命令分离出隐藏文件:

dd if=carter.jpg of=carter-1.jpg skip=140147 bs=1

可以参考。dd命令详解,这里if是指定输入文件,of是指定输出文件,skip是指定从输入文件开头跳过140147个块后再开始复制,bs设置每次读写块的大小为1字节。

最后我们可以得到这样的一张carter-1.jpg图片:



(2) 使用foremost工具分离

foremost是一个基于文件文件头和尾部信息以及文件的内建数据结构恢复文件的命令行工具,win可以下载地址,Linux可以

通过下面命令安装使用:

foremost carter.jpg

当我们使用这行命令后,foremost会自动生成output目录存放分离出文件:

< > 企 主文件夹 桌	面 output jpg		Q ⅲ ≡ ⊜ ⊕ €
③ 最近使用的			
🔂 Home	1	100	
■ 桌面	00000000.jpg	00000273.jpg	
日 祝頻			
◎ 图片			
□ 文档			
④ 下载			
♬ 音乐			
💮 回收站			
+ 其他位置			

(3) hex编辑器分析文件

至于hex编辑器有很多,win下有用得较多的winhex,UltraEdit等,linux下有hexeditor等,这里我们以winhex为例手动分离,在分离之前我们需要知道一点关于jpg文件格式的知识,jpg格式文件开始的2字 节是图像开始SOI(Start of Image,SOI)为FF D8,之后2个字节是JFIF应用数据块APPO(JFIF application segment)为FF E0,最后2个字节是图像文件结束标记EOI(end-of-file)为FF D9,如果你想详细了解 更多关于这方面的知识可以参考jpg文件格式分析一文。

用winhex打开图片,通过Alt+G快捷键输入偏移地址22373跳转到另一张jpg的图像开始块,可以看到FF D8图像开始块。



而图像结束块FF D9

ase Data	0 🗳 🖬 🍳		E.	5		5		3	ß	B 01	2	ê	4	-	25	H		→ +£1 🗢 → 🛛 🍣	2
le E <u>d</u> it	carter.jpg																		
	Offset	0	1	2	3	4	5	6	7	8	9	A	в	С	D	Ε	F		,
	0002F4A0	E9	67	CA	93	96	96	4F	9A	51	80	9A	49	59	C5	72	B 8	égÊ"CšQŒšIYÅr	
	0002F4B0	BB	C9	92	F7	F5	A9	AD	DB	6F	CD	39	49	D9	5E	ED	34	»É'÷õ€-ÛoÍ9IÙ^i	4
	0002F4C0	A5	28	A9	5D	BD	75	BA	49	2F	DE	AF	D8	57	F6	67	FO	¥(@]hu°I/1 @Wög	5
	0002F4D0	4F	C3	EF	80	5F	B1	7E	A9	1E	9F	E1	DD	5F	50	B 9	BD	CĂIC ±~C YáÝ P	5
	0002F4E0	F1	E7	C7	1F	13	F8	D6	4D	19	2D	75	8D	67	48	B4	79	ñçÇ gÖM -u gH'	v
	0002F4F0	53	4A	99	D9	B 3	2C	71	48	A6	29	85	BB	BE	D4	77	EC	SJ™Ù*,qH;)»ħĈw	i l
	0002F500	CA	4B	7C	81	E1	OF	89	7A	67	81	3F	EO	8D	3F	B4	A7	ÊK á tizg ?à ?'	5
	0002F510	C5	OF	88	1A	34	9E	21	BA	FD	AD	3E	24	47	A4	F8	6B	Å * 4ž!°ý->\$G#@	ĸ
	0002F520	4E	5B	96	B 3	92	C2	EE	CO	DC	6A	12	EA	B 2	4D	19	26	N[-"'ASAUJ &*M	6
	0002F530	68	E3	9A	ED	50	40	C3	6B	3C	79	27	OD	91	F9	A9	5E	hāšiP@Āk <y' 'ù@<="" td=""><td>^</td></y'>	^
	0002F540	FS	FB	59	7F	C1	42	FC	59	FB	5B	FC	20	F8	5F	EO	OB	ôùY ÁBUYÛ[Ü @_à	
	0002F550	EF	0E	78	OF	C1	7E	0C	F8	4D	A7	3D	9E	91	A4	78	45	I x Å~ eMS=2 'Hx	0
	0002F560	49	6D	3A	DE	EA	79	44	62	7B	FB	AO	D2	48	65	BB	9B	Im: PêyDb(û ČHe»	>
	00027570	CB	43	24	99	50	C7	27	68	2C	C4	E7	52	9D	E9	4A	90	ECS"PC'h, AcR éJ	æ
	0002F580	76	69	45	2E	CB	9E	32	6F	4E	BC	94	E9	D3	56	49	A7	VIE. Ež2oN4"éÓVI	s
	00027590	1E	75	66	DD	B4	8D	46	EA	F3	CF	AB	72	6F	BB	4A	56	ufý' Féóľ«ro»J	v
	0002F5A0	5E	9E	D2	A4	E6	EE	DA	E5	F 7	1D	D2	D7	C1	68	A2	8A	^žÔ¤ælÚå÷ Č×Áho	ŝ
	0002F5B0	DO	cc	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	ĐÌ (oš (oš (oš (0
	0002F5C0	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	Š (oŠ (oŠ (oŠ (0
	0002F5D0	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	š (oš (oš (oš (0
	0002F5E0	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	š (oš (oš (oš (0
	0002F5F0	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	š (oš (oš (oš (0
	00027600	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	SA	00	28	A2	š (oš (oš (oš (0
	0002F610	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	Š (0Š (0Š (0Š (0
	0002F620	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	š (oš (oš (oš (0
	0002F630	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	Š (oŠ (oŠ (oŠ (0
	0002F640	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	š (oš (oš (oš (0
	00027650	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	Š (OŠ (OŠ (OŠ (•
	00027660	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	Š (oŠ (oŠ (oŠ (0
	00027670	8A	00	28	82	8A	00	28	A2	8A	00	28	A2	8A	00	28	A2	š (oš (oš (oš (0
				_	_											-	-		

选取使用Alt+1快捷键选取FF为开始的块,Alt+2选取D9为结束块,然后右键->Edit->Copy Block->Into New File保存相应的文件后缀,例如new.jpg



还有一种特例,它是事先制作一个hide.zip,里面放入隐藏的文件,再需要一张jpg图片example.jpg,然后再通过命令 copy /b example.jpg+hide.zip output.jpg生成output.jpg的新文件,原理是利用了copy 命令,将两个文件以二进制方式连接起来,正常的jpg文件结束标志是FF D9,而图片查看器会忽视jpg结束符之后的内容,所以我们附加的hide.zip就不会影响到图像的正常显示。(参考AppLeU0的 *隐形* 式总统)

针对这种特例我们可以直接将jpg文件改为zip文件后缀(其他文件如rar文件也类似),就可以看到hide.zip压缩包里隐藏的文件。

比如当我们得到一张wh3r3_is_f14g.jpg文件:



当我们用winhex打开文件,发现wh3r3_is_f14g.jpg文件最后数据块不是FF D9 jpg文件的结束标志,而是zip文件的结束标志。

Case Data	0 🖪 🖬 4	30	E.	2		5		3	ß	B 10	2	â	-	-	25	HD		→ + + +	3
File Edit	wh3r3_is_f14g	jpg																	
	Offset	0	1	2	3	4	5	6	7	8	9	A	в	С	D	Ε	F		^
	00027AD0	AA	D4	4C	6A	B 2	50	E5	34	03	CF	5F	E4	F9	15	90	3D	*ÔLj*På4 Ϊ äù α=	
	00027AE0	1A	7E	D3	1B	6D	30	94	AF	F4	FC	C6	FF	7B	1F	72	61	~Ó m0~ ôüÆy{ ra	
	00027AF0	15	A8	08	19	3D	76	10	10	CB	14	43	64	87	C7	26	39	- =v È Cd‡Ç69	
	00027800	39	EE	A1	1E	0A	8E	37	70	A0	EA	AO	7F	95	D6	6D	62	91; Ž7 ê •Čmb	
	00027B10	CO	11	D3	85	98	D8	9D	30	E6	4B	48	cc	6C	91	11	91	À Ó "Ø OæKHİl' '	
	00027B20	15	82	52	A0	49	29	05	36	E3	60	AE	89	AB	C8	08	77	,R I) 6ā'StaÈ W	
	00027B30	F1	B 5	AA	DC	31	C6	62	05	08	A4	47	3C	43	50	65	FO	ñµ=ÜlÆb ¤G <cfeð< td=""><td></td></cfeð<>	
	00027840	81	78	49	44	E4	BO	87	F4	18	E5	C2	3F	2E	2F	2F	13	xIDă°‡ô åÂ?.//	
	00027850	D1	57	SF	7D	B 5	BO	BO	B 0	B 9	B 9	B 9	B 2	B 2	B2	B 7	6B	NW)µ*********k	
	00027860	1E	3F	7E	5C	6B	D4	3A	9D	4E	E3	54	DA	6E	B 7	93	24	?~\kĈ: NāTÚn ·"\$	
	00027B70	71	CE	D5	6A	B 5	5A	AD	66	80	39	7D	FA	34	E2	BA	2C	qîõjuZ-f@9)ú4á°,	
	00027880	4E	96	4D	BE	34	56	4C	B 3	C2	79	OF	1F	OA	C7	31	90	N-MM4VL*Åy Ç1	
	00027890	2F	10	72	BF	B 0	3C	ΒE	99	6E	74	FO	8B	1F	61	07	87	/ r¿°<%"ntőc a ‡	
	00027BA0	D7	97	48	D3	2B	C8	96	85	OD	31	84	C2	73	18	AD	30	*-HÓ+È 1"Ås -<	
	00027BB0	93	59	C9	AE	28	63	62	11	8F	86	8F	FA	B1	45	B1	6E	"YÉS(cb † ú±E±n	
	00027BC0	10	68	A3	75	A0	94	03	C6	58	F1	02	00	A5	C4	F1	C6	h£u " #Xñ ¥ĂñÆ	
	00027BD0	70	44	cc	7E	52	FO	7 F	0C	02	1E	ED	AF	74	60	10	00	pDI~R8 it.	
	00027BE0	00	00	00	49	45	4E	44	AE	42	60	82	50	4B	03	04	0A	IENDSB', PK	
	00027BF0	00	00	00	08	00	E2	A1	A8	48	F2	4E	19	2D	1D	00	00	á; HôN -	
	00027C00	00	1B	00	00	00	08	00	00	00	66	6C	61	67	2E	74	78	flag.tx	
	00027C10	74	4B	CB	49	4C	AF	0E	2E	31	4E	37	C9	33	48	2F	32	tKËII ⁻ .1N7É3H/2	
	00027C20	29	C8	A8	8C	CF	2C	8E	48	2B	CD	53	AC	05	00	50	4B)È"ŒĨ,ŽC+ÍS- PK	
	00027C30	01	02	0A	00	0A	00	00	00	08	00	E2	A1	AB	48	F2	4E	á; "HòN	
	00027C40	19	2D	1D	00	00	00	1B	00	00	00	08	00	00	00	00	00	-	
	00027C50	00	00	00	00	20	00	00	00	00	00	00	00	66	6C	61	67	flag	
	00027C60	2E	74	78	74	50	4B	05	06	00	00	00	00	01	00	01	00	.txtFK	
	00027C70	36	00	00	00	43	00	00	00	00	00							6 C	
																			~
	Page 364 of 3	64					Offe.	ot-			2703	76					= 0	Block	

我们直接将文件改名为wh3r3_is_f14g.zip,打开得到flag.txt。



最后打开flag.txt得到flag。

■ flag.txt - 记事本	-	×
文件(F) 編編(E) 格式(O) 查看(V) 帮助(H)		
flag{St3g4n0gr4phy_is_fun!}		^

0x03 后话

图片隐写方式有很多种,在此只介绍了这一种,如果以后有机会会写其他的图片隐写,如果对隐写感兴趣这里推荐一本机械工业出版社的《数据隐藏技术揭秘:破解多媒体、操作系统、移动设备和网络协议中的隐秘数据》,如果你不想购买实体书,可以*下载pdf版*。