CTF基础-图片隐写篇



2 篇文章 1 订阅 订阅专栏

0x0前言:本人作为一个**ctf**菜鸟,在学习的过程中遇到了很多的疑问,大多数时候都是 通过百度或谷歌解决。但由于没有一个整合的帖子,所以很多资料都十分零散,为了自 己能方便浏览,也为了能方便更多新手,于是决定写下这篇文章。错误的地方欢迎大家 指正。

【工具及题目链接:https://pan.baidu.com/s/1up969RLLbDP0Wlky4QdSew 提取码:f62y】

0x1常见隐写类型:

【以下题目多来源于各大ctf题库或网站,仅修改了文件名以便于讲解】

1.利用binwalk工具分离图片

2.stegsovle工具的利用

3.txt简单隐写

4.关键字搜索

5.十六进制文件头补全及修改

6.png格式IHDR的问题

7.属性隐写+文件类型



binwalk+steghi de+属性隐写. jpg



关键字搜索



binwalk分离.jpg



十六进制文件头(补全).jpg





stegsovle.png





stegsovle2.png

ez.png tx



十六进制文件头(

修改).jpg

https://blog.csdn.net/Dog_Captain

0x2常用工具:

1.kali虚拟机【binwalk、foremost】

2.十六进制编辑器【Winhex或Hexedit】

3.记事本或其他文本编辑器

4.stegsovle【基于java运行,需配置java环境】

1x0题目详解:

1x1利用binwalk工具分离图片:

1.首先将文件放到kali下,利用binwalk命令查看图片,通过查看描述可以发现图片中还隐藏了另外一张图片。

2.接着利用binwalk -e或foremost命令来分离他们。

root@kali:~/桌	面 / ctf练 习 # bin v 一 六 进 制 文 一 一 六 进	Walk binwalk分离.jpg 問文
DECIMAL	HEXADECIMAL件头()	DESCRIPTION
0	0x0	JPEG image data, EXIF standard
12	0xC	TIFF image data, big-endian, offset of first image
directory: 8		
13017	0x32D9	Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:descript< td=""><td>tion rdf:about="'</td><td><pre>' xmlns:photoshop="http://ns.adobe.com/photoshop/1.</pre></td></rdf:descript<>	tion rdf:about="'	<pre>' xmlns:photoshop="http://ns.adobe.com/photoshop/1.</pre>
0/" xmlns		
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
164186	0x2815A	Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:descrip< td=""><td>tion rdf:about="'</td><td>' xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns</td></rdf:descrip<>	tion rdf:about="'	' xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns
:xap="htt		
168370	0x291B2	Copyright string: "Copyright (c) 1998 Hewlett-Pack
ard Company"		https://biog.csun.nei/Dog_Capiain

<mark>root@kali</mark>:~/桌面/ctf练习# foremost binwalk分离.jpg Processing: binwalk分离.jpg |*|

falg{	7}
	https://blog.csdn.net/Dog_Captain

3.在生成的目录中,我们可以发现被隐藏的图片,从而得到flag。

1x2stegsolve工具的利用:

1.首先打开stegsolve,并open题目图片

StegSolve 1.3 by Caesum	—		\times
File Analyse Help			
Normal Image			
https://blog.	csdnine	t/Dog_C	aptain

2.通过点击左右箭头,来查看图片在其它文件格式下的图像,然后在Gray bits格式下,发现一张隐藏的二维码, 扫描二维码即可得到flag



3.在其它格式下也可能会有提示信息,例如这张图



4.在Green plane0格式下直接发现了flag



1x3 txt简单隐写

【此类题目可以说是最简单的一类题,通过直接使用记事本或者其它文本编辑软件,可直接查看。一般拿到图 片后,大多先用记事本查看,从而判断是否为txt简单隐写。一般flag会位于文本开头或结尾,少数情况会在文本 中间,这类情况后面会单独介绍】

1.拿到图片后,直接用记事本打开,在文本结尾发现了一串编码,学习过密码学的人应该很容易认出这是 Unicode加密。我在之前的帖子里,也介绍过这种编码方式 【https://blog.csdn.net/Dog_Captain/article/details/82690338】



2.利用解码器可直接解flag

(1Ŧ	复制/粘贴	过滤器	格式	统计	工具 扩展	
					输入(面换值)	
转	换选项 ———				x#107.x#101.x#121.x#123.x#121.x#111.x#117.x#32.x#97.x#114.x#10	1
	Text to Hex	He	x to Text		1, ,r,i,g,h,t,}	· · ·
	Dec to Hex	He	x to Dec			
	Text to Dec	De	c to Text			
	Dec to Octal	Oct	al to Dec			
	Text to UTF7	UTF	7 to Tex	t I		
	Hex to UCS2	UCS	52 to Hea	<u> </u>		
	Text to Binary	Bina	ry to Tex	t		
	Escape	Un	nescape			
	Encode HTML	Deco	ide HTM			
	Text to Base64	Base	64 to Te	xt I		
	Hex to Base64	Base	e64 to He	×		
_				_		
	SHIFTx 拆分所有 拆分所有 保留所有		 字符, Delim 行		输出转换值): key ^r	^
Ē	1	2 2	12.99	_		
1	└ 子付·所有	14	12五			
	开始位置	1				
	S	wap				
Γ	1 字符.所有 开始位置	2 1 1	位置 盾环 [1	-		
				1		
	将输出复	制到剪厕	站板	_		
	将输出复 复制输	制到剪「 出到输)	贴板 入			
	将輸出复 (夏制輸 全部	(制到剪) (出到输)))消除	贴板 入			~

1x4 关键字搜索

【这类题目与txt简单隐写类似,但flag大多隐藏在文本中间,当文本过长的时候,只靠肉眼查找会浪费很多时间,我们可以直接用ctrl+f搜索关键字,例如:flag、FLAG、key、KEY或根据赛方要求的格式进行查找】

💭 关键字搜索 - 记事本		- 🗆 X
文件(E) 编辑(E) 格式(O)) 查看(<u>V</u>) 帮助(<u>H</u>)	
□[]A\A]A^A	? 竺 H泼□H兡□? □ <mark>flag</mark> { } welcome to the re world □□	⊐;4 □ 瘕 ^
€@?P-	? P ? ? □ □ □zR □x□□□_▲□□?□□□ 椠 *	□ □zR
oxooo∧oo? \$	h? P ==F==J==w=€ ?=;*3\$" D }? A==?C	
□Z♠□□ D d €?	e Baa?Eaa?Ea ?Ea(?Ha0?Ha8?Ma@la8Aa0Aa(Ba BaaBaaBaa a	?
2	5 ^比 ✓	
<u>.</u>		
7	查找内容(N): flag	
-		
	方向取消	
	_ 区分大小写(<u>C</u>)	
5	□循环(R)	
@ _		2@
		:@
	₩ U L ? U XUW ` V_@ f_@ v_@ 2@ f_@ CCC:////	
		untu 4.8.4-
2ubuntu1~14.04)	4.8.4 GCC: (Ubuntu 4.8.2-19ubuntu'l) 4.8.2 .symtab .strtab .shstrtab .in	terp ~
从顶部找到的下一步	Macintosh (CR) 第1行,第1465例2	100%nevDog_Captain

1x5十六进制文件头补全及修改

【首先,我们需要知道文件头是位于文件开头,用处承担一定任务的数据,我们可以通过文件头来判断文件类型。因此,当文件头被删除或修改后,文件可能会打不开,这种情况我们就要根据所给文件的后缀,补全或修改文件头】

1.先观察所给的文件类型,发现是jpg格式,我们就要想到jpg文件的文件头为FFD8FF



2.利用Winhex打开图片,查看文件的十六进制,观察后我们会发现,相比于正常的jpg文件,这张图的文件头缺失了三位

```
十六进制文件头(补全).jpg
```

Offset	0	1	2	3	4	5	6	7	8	9	А	В	С	D	Е	F		
00000000	E0	00	10	4A	46	49	46	00	01	01	00	00	01	00	01	00	à JFIF	
00000010	00	$\mathbf{F}\mathbf{F}$	DB	00	43	00	02	01	01	01	01	01	02	01	01	01	ÿÛ C	
00000020	02	02	02	02	02	04	03	02	02	02	02	05	04	04	03	04		
00000030	06	05	06	06	06	05	06	06	06	07	09	08	06	07	09	07		
00000040	06	06	08	0B	08	09	0A	0A	0A	0A	0A	06	08	0B	0C	0B		
00000050	0A	0C	09	0A	0A	0A	\mathbf{FF}	DB	00	43	01	02	02	02	02	02	ÿÛ C	
00000060	02	05	03	03	05	0A	07	06	07	0A	0A	0A	0A	0A	0A	0A		
00000070	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A		
00000080	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A		
00000090	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	0A	$\mathbf{F}\mathbf{F}$	C0	00	11	08	/blog.csdp.net/Do ^{ÿÀ} Capta	
000000A0	04	38	04	38	03	01	22	00	02	11	01	03	11	01	FF	C4	88" VÄ	

3.将文件头补全后保存,系统会自动生成一个备份文件,防止修改错误,而原文件则会变为正常图片

FF	D8	FF	ΕØ	00	10	48	46	49	46	00	01	01	00	00	01	<mark>ÿ0ÿὰJ</mark> FIF
00	01	00	88	FF	DB	00	43	00	02	01	01	01	01	01	62	ij?C
01	01	01	02	02	02	02	02	64	03	02	02	02	02	05	64	
64	03	04	86	05	86	86	86	05	86	06	86	07	89	08	86	
07	09	07	06	06	08	OB	08	89	ØA	ØA	ØA	ØA	ØA	06	68	
ØB	0C	ØB	ØA	9C	89	ØA	ØA	ØA	FF	DB	00	43	01	02	82	ü?(
02	02	02	62	05	03	03	85	ØA	07	06	07	ØA	ØA	ØA	ØA	
80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	



十六进制文件头(补全).jpg



4.文件头修改同理,观察文件头前几位,将其修改为正确格式,保存后即可得到正常图片





1x6 png格式的IHDR问题

【这类题目最大的标志:图片格式为png,或当你感觉这张图片好像被裁掉一部分时,要考虑的这类问题。这类题目的原理我理解的并不是很深刻,仅仅局限于解题方法...】

1.先查看一下原图



2.用十六进制编辑器打开图片后,我们会发现他的标志IHDR,对应左边十六进制的49 48 44 52,我们以此为界,后面的四位为图片宽度,再向后四位为图片高度

Ī	89	50	4E	47	ØD	ØA	18	ØA	00	00	00	ØD	49	48	44	52	%PNGIHD
	00	00	02	87	00	00	01	00	08	06	00	00	00	6D	70	7振	⊭?m q
1	35	ុព្	00	00	01	73	52	47	42	00	AE	CE	10	E9	00	0011	5sRGB?.
1	00	64	67	41	4D	4 平	66	00	B1	8F	OB	FC	61	05	00	00	gAMA睆.黙
1	00	09	70	48	59	73	00	00	ØE	64	00	00	ØE	C4	01	95	pHYs???
l	2B	0E	1 B	00	00	FF	A5	49	44	41	54	78	5E	EC	BD	67	+ij DATx^旖.
1	A Ø	A5	57	59	EE	FF	EE	BE	4F	9B	DE	93	40	78	ØF	84	牓₩Y?罹O涋揕z.?
1	24	24	60	9C	04	A5	2B	20	45	10	10	BB	88	8A	A 8	57	\$\$`? E粓姩W
1	BD	FC	EF	BD	78	F5	5A	AE	7A	BD	5E	CB	BD	2A	62	05	近锝z鮖畓絕私*b.
l	04	69	52	64	E9	01	42	48	48	42	7A	EF	7D	52	A6	CF	.iR.?BHHBz食TR○
	9C	7E	76	FD	3F	BF	F7	DB	EF	39	6B	76	Fő	40	26	69	□猫u?亏埏9kυ鯨&?

2.接下来,我们根据实际情况,将宽度和高度改为相同数值,然后保存

8	9	50	4E	47	ØD	ØA	18	ØA	00	00	00	ØD	49	48	44	52	
0	0	00	02	A7	00	00	02	A7	08	86	00	00	00	6D	70	71	§§m
3	5	00	00	00	01	73	52	47	42	00	AE	CE	10	E9	00	66	5sRGB?.
0	0	04	67	41	4D	41	00	66	B1	8F	6B	FC	61	05	00	66	gAMA睆.黙
0	0	09	70	48	59	73	00	66	0E	C4	00	00	0E	64	01	95	pHYs???
2	B	ØE	1 B	88	00	FF	A5	49	44	41	54	78	5E	EC	BD	07	+j DATx^旖.

3.再次查看图片,发现隐藏的flag



1x7 属性隐写+文件格式

详情请见我写过的题目解析【https://blog.csdn.net/Dog_Captain/article/details/84567858】

2x0 小结

以上,就是我所遇到的常见的基础隐写题目,在各大ctf比赛中,直接出现的机率较小,即使出现,分值一般也 较低。大多数情况都是结合密码题或其它类型的题目一同出现,所以仅仅掌握这些常见的隐写方式只是基础。 希望各位初学者不要以为掌握了这些题目就可以驰骋ctf赛场,前面的路还很长(笑)。

最后,谢谢大家的阅读与学习~