CTF—图片隐写+数据隐写



<u>小常吃不下了</u> <u>小常吃不下了</u> <u>+ 2021-07-28 17:27:29 发布</u> <u>549</u> <u>w</u>藏 1 分类专栏: <u>CTF</u> 文章标签: <u>安全 unctf</u> 版权声明:本文为博主原创文章,遵循 <u>CC 4.0 BY-SA</u> 版权协议,转载请附上原文出处链接和本声明。 本文链接: <u>https://blog.csdn.net/weixin_52620919/article/details/119184050</u> 版权



CTF 专栏收录该内容

32 篇文章 1 订阅 订阅专栏

一、【图片隐写】

题目描述:

```
在实验主机上的C:\Stegano\3目录下提供了一个名为stego的文件,
找到一个形式为flag{word_word_word}的字符串。
```



预备知识

[TrID]

TrID是一款根据文件二进制数据特征进行判断的文件类型识别工具。 虽然也有类似的文件类型识别工具,但是大多数都是使用硬编码的识别规则,而TrID则没有固定的匹配规则, TrID具有灵活的可扩展性,可以通过训练来进行文件类型的快速识别。

TrID通过附加的文件类型指纹数据库来进行匹配,可用于 取证分析、未知文件 识别等用途。

[BinWalk]

```
BinWalk是一个固件的分析工具,旨在协助研究人员对固件进行分析,提取及逆向工程用处。
简单易用,完全自动化脚本,并通过自定义签名,提取规则和插件模块,
还有重要一点的是可以轻松地扩展。最简单的使用方法很直接,提供文件路径和文件名即可。
```

[StegHide]

```
Steghide是一个隐写程序,其可以将数据隐藏在各种图片文件以及音频文件之中。
Steghide可以对隐写的数据进行加密和压缩操作。
```

考察意图:

考察选手的文件隐写取证分析能力,包括对工具TrID、BinWalk、StegHide等的了解。

分析文件类型

题目提供的文件没有文件扩展名,因此我们需要先确定文件的真实文件类型, 使用【TrID】工具可以对未知文件类型进行有效的识别。

1、打开cmd命令提示符,切换到相关路径下,然后使用TrlD对文件进行识别,如图所示:



根据TrID的识别结果,这是一个GZip压缩文件,我们使用7Zip对其进行解压 得到另一个没有扩展名的文件,再次使用TrID对其进行识别,提示为TAR压缩文件,如图所示:



同样使用7Zip对其进行解压,得到 bowser.jpg 文件,是一张图片。



内嵌文件数据分析

TPEG 图像

在CTF竞赛中,很多情况下会把一个文件的二进制数据嵌入到另一个文件之中, 对于[嵌入ZIP]之类的数据,我们可以尝试直接将文件名的后缀改为zip, 然后使用WinRAR之类的解压缩工具打开即可 但是对于未知的文件类型,这样可能就显得力不从心了,这里介绍一种通用的文件识别方法,就是使用BinWalk工具。 打开cmd命令提示符,切换到相关路径下。执行python binwalk命令来对bowser.jpg文件进行处理,如图所示:

🔤 命令提示	符		- 🗆 🗙
C:\Python2	7\Scripts>python	binwalk C:\Stegano\3\bowser.jpg	_
DECIMAL	HEXADECIMAL	DESCRIPTION	
140981 ssed size: 182338	0x226B5 41199, uncompres 0x2C842	Zip archive data, at least v2.0 to extract, sed size: 42877, name: MarioCastle.jpg End of Zip archive	compre

从BinWalk的分析结果可以看出,JPG文件中有一个Zip压缩包,其文件偏移地址范围是0x226B5~0x2C842,这里我们使用 C32Asm将Zip数据提取出来。

打开桌面上的【C32Asm工具】,选择"文件"、"打开十六进制文件"载入C:\Stegano\3\bowser.jpg文件,然后右键选择"定义选择块" 填入数据块的起始地址为 0x226B5,结束地址为 0x2C842,单击确定就选中数据块了 右键复制数据,然后新建一个十六进制文件,将原有的数据替换为复制的数据 保存即可得到压缩包文件。操作过程如图所示:

● 文件(E) 编辑(E) 搜索(S) 查看(V) 工具(T) 高級(A) 窗口(W) 帮助(H) ●	🚺 – [(нех	() bor	/ser.jpg]										
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●	🙀 文件(E)	编辑	:(<u>E</u>) 搜索(<u>S</u>) 査	看(⊻) 工具(I	5 (高级(<u>A</u>)	窗口	⊐(<u>W</u>)	帮助(<u>H</u>))		
Enjoy C32am (HEX)bowser.jpg 600000001 c: CE D0 CE C0 00 10 10 10 10 10 10 10 10 10 10 10 10	i 💕 🖬 🏼	1 🕰	5,	5 3 9	1	*= *=	>			** 12	3 B\$		
8080808081 FE De GE DE FE DE GE DE FE DE GE DE DE <thde< thd=""> DE <thde< thd=""></thde<></thde<>	🔛 Enjoy C32a	asm 🛔	🙀 (HEX)bowser.j	jpg									
00000010: 拷贝(C) ▶ 08 06 07 06 05 08 .Hÿ?C 00000020: 粘貼(P) Ctrl+V 00 0C 08 06 07 06 05 08 .Hÿ?C 00000030: 剪切(T) Ctrl+V 00 0C 08 06 70 06 05 08 .Hÿ?C 00000030: 剪切(T) Ctrl+X 30 31 34<	00000000:	ce i	0 EE EA AA 4	ց հծ հե հն	46	00	01	02	<mark>01</mark>	00 48	ij??.JF	·IF	. H
000000020: 粘贴(P) Ctrl+V 00 0C 0B 0F 定义选择块 000000030: 剪切(T) Ctrl+X 30 31 34 34 000000060: 全选(A) Ctrl+A 21 1C 21 32 00000060: 定义选择块(D) 00 02 02 22 45 00000000: 定义选择块(D) 00 02 02 22 45 0000000: 定义选择块(D) 00 02 02 02 05 0000000: 定义选择块(D) 00 02 02 02 05 00000: 00 02 02 02 05 00 02 02 02 05 文件开始 00 02 02 02 05 文件开始	00000010:		拷贝(<u>C</u>)	•	08	06	06	07	06	05 <mark>08</mark>	.Hÿ?	20	
000000000000000000000000000000000000	00000020:		粘贴(<u>P</u>)	Ctrl+V	ØD	00	ØB	ØF	τ 定 Ψi	法择世			
090000050: 全选(A) Ctrl+A 30 31 34 31 块开始 090000060: 全选(A) Ctrl+A 21 12 132 0x226B5 < 文件开始 090000000: 定义选择块(D) 0x 20 02 02 02 05 yt4束	00000030:		煎切(工)	Ctrl+X	10	10	20	24	~~~				
999090969: 全选(A) Ctrl+A 21 10 0x226B5 < 文件开始 ▼ 999090909: 定义选择块(D) 0x206B5 文件开始 ▼ ●	00000040:				30	31	34	34	快刊	千始			
999000097 9: 定义选择块(D) 22 23 0 0x22005 < 文件升始 ▼ 90900097 9: 21 0 22 25 0 0x22005 < 文件升始 ▼ 90900097 9: 9090000 9 20 00 0x22005 < 文件升始 ▼	00000050:		全选(<u>A</u>)	Ctrl+A	94	10	91	44		1.24	and 4 -	- /14 -77 4.7.	
	666666676:	Г	定义选择块(D)		21	99	21	34			000 X X	计开始	-
	00000080:	-		•	32	32	32	32	块约	舌 鬼			
9999999999: 32 32 32 32 0x2C842 < 文件开始 ▼	00000090:		1550X 500 1550 (<u>M</u>)		32	32	32	32		0x2C	845 < 文	件开始	-
9999996A9: ^{填充(E)} 22 99 92 11	000000A0:		填充(E)		22	00	02	11					
000000B0: 協協(山) Ctrl.7 01 01 01 01	000000B0:		物時的目的	Ctrlu Z	01	01	01	01					
【099090009: 1101110 CUT-2 03 04 05 04 □光标定位于块的开始(CP	0000000000			Cui+z	03	04	05	96	÷ 🗆	光标定位 ⁻	于块的开始	\$ (C,	
●999990009: 車做(R) Ctrl+Y 91 93 93 92 (6.位置定位(D) 确定(O)	000000D0:		重做(<u>R</u>)	Ctrl+Y	01	03	03	02	6	位罢守位	(p)	确。	È(0)
19999999E 9: 19999999 5: 11 単安(F) (the solid of states)	000000E0:	65	相表(5)	CtrluE	03	00	04	1		区里花区	(<u>r</u>)		
14 32 81 97 (天小定位(S)) 退出(C)	000000100			Curri	14	32	87	4	blog	天小定位	seveixir	退退	<u>Ц(C)</u>

但是实际测试时发现压缩包不能正常解压,这是因为BinWalk对ZIP的识别机制还不完善,漏掉了ZIP文件末尾的一段数据,所以 我们需要重新复制数据,文件的开始位置仍然是0x226B5,结束位置直到文件的末尾,如图所示:

定义选择块 🔀
块开始
0x226B5 < 文件开始 ▼
块结束
20858 < 文件结尾 🔻
● 位置定位(P) 确定(O)
○ 大小定位 (<u>S</u>) 退出 (C)

这样提取出来的压缩包就可以正常解压了,得到图片文件MarioCastle.jpg



提取隐写数据

从实验步骤二中得到了MarioCastle.jpg文件, 这个图片里面显示有一个Passphrase字段,值为BaD_DR4G0N, 看到 [Passphrase]我们就可以猜测使用了【steghide】进行数据隐写。 因此我们尝试使用steghide对MarioCastle.jpg进行隐写数据提取



打开cmd命令提示符,输入steghide extract -sf MarioCastle.jpg -p BaD_DR4G0N命令,提示没有任何可以提取的数据。 那么加里尝试对原来的 bowser ing 进行处理呢? 实际测试表明bowser ing 用面隐写了数据,通过steghide我们提取出

那么如果尝试对原来的 bowser.jpg 进行处理呢? 实际测试表明bowser.jpg里面隐写了数据,通过steghide我们提取出了1.tar.gz文件,如图所示:



解压1.tar.gz文件,得到flaga.jpg文件,打开即可看到flag为flag{You_F0unD_M3}。