图片隐写 安恒ctf_图片隐写 安恒ctf_CTF中图片隐写的一些整理总结

原创

Ms 汪 于 2021-02-05 08:45:27 发布
文章标签： 图片隐写 安恒ctf

对历年来国内外CTF中常见的题型图片隐写的一些总结，本文长期更新，及时补充新的题型。

对历年来国内外CTF中常见的题型图片隐写的一些总结

赛题

XMAN-qualifiers-2017 : Misc/SimpleGIf

考察点Gif文件头

隐写相关技巧

Python 脚本编写

工具010Editor

Linux identify命令

Visual Studio Code

附件

Writeup

GIF头补全

首先用file查看下这个文件:

ctf@ubuntu:~/Desktop$ file '/home/ctf/Desktop/100_KHf05OI.gif'

/home/ctf/Desktop/100_KHf05OI.gif: data

binwalk再看下:

ctf@ubuntu:~/Desktop$ binwalk '/home/ctf/Desktop/100_KHf05OI.gif'

DECIMAL HEXADECIMAL DESCRIPTION

可以看出没有识别出什么文件，这里推测它就是一个gif文件，现在来补上gif头。

GIF文件头：47 49 46 38 39 61

得到Gif，什么都没有发现，看来题目没有这么简单:

分析Gif

使用identify命令去拆解Gif，发现每一帧存在规律。

ctf@ubuntu:~$ identify -format "%s %T \n" '/home/ctf/Desktop/100_KHf05OI.gif'

0 66

1 66

2 20

3 10

4 20

5 10

6 10

7 20

8 20

9 20

10 20

11 10

...

提取每一帧的间隔并进行转化：

ctf@ubuntu:~$ identify -format "%T\n" '/home/ctf/Desktop/100_KHf05OI.gif'

66

66

20

10

20

10

10

20

20

20

20

10

20

...

这里有2种可能：

20 & 10 分别代表 0 & 120 & 10 分别代表 0 & 1

20 & 10 分别代表 1 & 0

使用Visual Studio Code来进行批量替换:

去掉最前面2个66，一共是304行，304÷8=38 8个二进制为一组分组。

最后替换是这个样子:

数据处理这里20代表0，10代表1,写一个脚本来进行转换，二进制先转换为10进制，10进制转换为ascii码，asscii转换为相应的字符串。# coding:utf-8

blog:www.sqlsec,com

import re

def file():

text = raw_input("pleast enter your file path:")

with open(text,'r') as f:

binfile = (f.read()).replace("\n","")

bindata = re.findall(r'.{8}',binfile)

for i in bindata:

ascii = int(i,2)

print(chr(ascii))

file()

很明显flag是XMAN{96575beed4dea18ded4735643aecfa35}

总结

得了解常见的头文件的结构，学会分析Gif文件，对二进制数据要敏感。

赛题

XMAN-qualifiers-2017 : Pretty_CatPretty_Cat

工具exiftool

16进制编辑器

附件

Writeup

使用16进制编辑器

首先Windows下查看下图片的 exif 元数据信息:

WE1BTntVNWU=base64解码得到XMAN{U5e 可以明显的感觉到是半个flag，尝试使用16进制编辑器打开查看找到完整的base64 flag加密后的信息:

最后拿到flag是U5e_3x1ftoo1}

标准解法使用exiftool

Linux下直接使用exiftool工具查看:

ctf@ubuntu:~$ exiftool '/home/ctf/Desktop/cat_t1dzFZS.jpg'

ExifTool Version Number : 10.10

File Name : cat_t1dzFZS.jpg

Directory : /home/ctf/Desktop

File Size : 4.7 kB

File Modification Date/Time : 2018:01:18 22:59:18-08:00

File Access Date/Time : 2018:01:18 23:00:34-08:00

File Inode Change Date/Time : 2018:01:18 23:00:34-08:00

File Permissions : rwxrw-rw-

File Type : JPEG

File Type Extension : jpg

MIME Type : image/jpeg

JFIF Version : 1.01

Exif Byte Order : Big-endian (Motorola, MM)

X Resolution : 1

Y Resolution : 1

Resolution Unit : inches

Y Cb Cr Positioning : Centered

Copyright : WE1BTntVNWU=

Comment : XzN4MWZ0b28xfQ==

Image Width : 144

Image Height : 144

Encoding Process : Progressive DCT, Huffman coding

Bits Per Sample : 8

Color Components : 3

Y Cb Cr Sub Sampling : YCbCr4:2:0 (2 2)

Image Size : 144x144

Megapixels : 0.021

在图片元数据的Copyright和Comment种查找到关键信息。

总结

入门级别图片隐写之一。

下面介绍这种图片隐写题的制作方法，依然是使用exiftool。

假设flag为

sqlsec{th13_1s_1m4g3_stg}

拆成2部分为:sqlsec{th13_1s和_1m4g3_stg}

分别写入图片的copyright和comment

ctf@ubuntu:~$ exiftool -copyright="sqlsec{th13_1s" -comment="_1m4g3_stg}"
'/home/ctf/Desktop/cat_t1dzFZS.jpg'

1 image files updated

可以看到成功的更新的图片的元数据信息:

赛题

ZCTF：Whisper

考点图片隐写

文件切割

工具binwalk

Linux dd命令

Linux strings命令

Linux base64命令

16进制编辑器

附件

Writeup

解压里面是一个压缩包一个png图片，压缩包是加密的，先看下图片hint1.png再决定要不要爆破压缩包。

打开看到一张空白的图片:

使用图片隐写的必备工具Stegsplve工具查看下:

得到提示信息:不应该直接爆破压缩包，这种图片里面含有重要信息，很可能就包含了压缩密码。

使用16进制编辑器查看下这个图片文件，再末尾的时候发现了大量的字符串:

很多字符串，这里binwalk下，查看下切割文件的位置:

ctf@ubuntu:~$ binwalk '/home/ctf/Desktop/hint1.png'

DECIMAL HEXADECIMAL DESCRIPTION

---

0 0x0 PNG image, 1000 x 150, 8-bit/color RGB, non-interlaced

41 0x29 Zlib compressed data, default compression

5984 0x1760 StuffIt Deluxe Segment (data):
f+XOzsDjboavwNtZvpjWWtvZsDDOfLiesSbRNPEQBEDgroTMqVBQrNkfVpHAuSOndiiwXODNISCYZMNrRAC

67936 0x10960 StuffIt Deluxe Segment (data):
fNuJpDERDdlaOZNai+Coqla+DTgsEtMVsBjIAYltAPtSMrMkQQIJPpakhwWalKJotVVBWpqHOMZPAwtWsHBV+

140829 0x2261D StuffIt Deluxe Segment (data):
fvPMpvjpTCfKGkjxG+VxfTVLftfvLraEDKvtDbSdXoZVizG+cCDDPKd+XEilKqxvAfeWQtgCKXKWXtVAR+eQAegF

187232 0x2DB60 StuffIt Deluxe Segment (data):
fjpEVPsHTwrCroGzGnoLwjvYwiAvqxTueAuvuSXJglbrvoICIHJ+RPWsHZpthrYJBXgGdCPPsvfwliCeRnebbHbn

244309 0x3BA55 eCos RTOS string reference:
"ecosKtHYsvxlJRanOwiWWNYxriBWYzcOlEVoPxNzpeXReQcaAdVkqGhRkZcMEnYJOklurR+hWscMNYRtRJx

247553 0x3C701 StuffIt Deluxe Segment (data):
fnvX+GlGMKE+aiHXuKDWOHIHMXSLVdoKKosARHnTzsABOjVzWBlkdxdAspDRGliQJOuTwqpDAtkSxdggQqo

277260 0x43B0C COBALT boot rom data (Flat boot rom or file system)

464214 0x71556 StuffIt Deluxe Segment (data):
fBEIlSs7G4qlOURP8XckU4UJDEVod3DZPnQgkDoAUgAAAFIAAAACZ3NveUF3JUodMBgAlAAAAGQ3YjA1Nzli

465493 0x71A55 Unix path:
/UnIrK/DAKFS1q9Q/xBhjDw5ujSKDGF/F3JFOFCQCa1TQwXWnQggDgATwAAAE8AAAACAmpT2kJ3JUodMBg

560653 0x88E0D StuffIt Deluxe Segment (data):
fsjukotKpnhdlvnRikhqnMRtuowkNqvHtZdVjKkaTPQruHrLRKBSLfcsZYqDXJLccOwDcAbMxWdVidQLNZnJhxdL

644141 0x9D42D StuffIt Deluxe Segment (data):
fO+rQvhXLctsaRfWOjjSXPehzSbWJrfVsMXrTMrTaQIB+GxbVXDblxnckrwlhXkiGuOutNLlErRHScpoESJKjwPpu

658201 0xA0B19 StuffIt Deluxe Segment (data):
fpiHlbngTGOa+TlwTCsrudlapwKkHvknJhoZiBJTAMRWlvIlBTBErvBOwtoMQ+REHsupLhpPtPOokWWiYXDEc

确认切割的位置为5984

使用Linux下的dd命令来切割:

查看下切割出的gg_1的文件里面是什么:

ctf@ubuntu:~/Desktop$ cat gg_1

Sef+XOzsDjboavwNtZvpjWWtvZsDDOfLiesSbRNPEQBEDgroTMqVBQrNkfVpHAuSOndiiwXODNISCYZMNrRA

这是什么鬼~不过好像是传说中中的base64加密呀，使用Linux base64解密看下:

将解密后的文件存储为gg_2

然后再看下文件内容:

吃惊，怎么内容看起来更变态了：

先不要慌，使用binwalk命令再分析这个文件看看:

ctf@ubuntu:~/Desktop$ binwalk '/home/ctf/Desktop/gg_2'

DECIMAL HEXADECIMAL DESCRIPTION

206415 0x3264F RAR archive data, first volume type: MAIN_HEAD

206495 0x3269F bzip2 compressed data, block size = 900k

206636 0x3272C bzip2 compressed data, block size = 900k

206778 0x327BA bzip2 compressed data, block size = 900k

206917 0x32845 bzip2 compressed data, block size = 900k

207054 0x328CE bzip2 compressed data, block size = 900k

207191 0x32957 bzip2 compressed data, block size = 900k

207329 0x329E1 bzip2 compressed data, block size = 900k

207471 0x32A6F bzip2 compressed data, block size = 900k

207606 0x32AF6 bzip2 compressed data, block size = 900k

207743 0x32B7F bzip2 compressed data, block size = 900k

207881 0x32C09 bzip2 compressed data, block size = 900k

208017 0x32C91 bzip2 compressed data, block size = 900k

208156 0x32D1C bzip2 compressed data, block size = 900k

208296 0x32DA8 bzip2 compressed data, block size = 900k

208434 0x32E32 bzip2 compressed data, block size = 900k

208575 0x32EBF bzip2 compressed data, block size = 900k

208708 0x32F44 bzip2 compressed data, block size = 900k

208847 0x32FCF bzip2 compressed data, block size = 900k

208985 0x33059 bzip2 compressed data, block size = 900k

209122 0x330E2 bzip2 compressed data, block size = 900k

209261 0x3316D bzip2 compressed data, block size = 900k

209399 0x331F7 bzip2 compressed data, block size = 900k

209539 0x33283 bzip2 compressed data, block size = 900k

209676 0x3330C bzip2 compressed data, block size = 900k

209812 0x33394 bzip2 compressed data, block size = 900k

...

再尝试用binwalk来分离里面的数据:

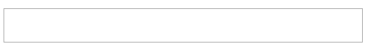
ctf@ubuntu:~/Desktop$ binwalk -e '/home/ctf/Desktop/gg_2'

解压出了一堆文本文件：

使用Linux下的strings命令来搜索文件夹内中文本内容里面所含的关键字:

找到了解压密码，用这个密码去解压压缩包看看(还好当初没有无脑爆破~)

解压出一个flag.txt文件，里面放着flag



ZCTF{Nightingale}

总结

熟练使用Linux下的一些自带命令如:strings，dd，base64等命令，在CTF比赛中可以提高解题的效率，不过这一题还真是够变态的。