

CTF misc图片类总结（入门级）

原创

[Hardworking666](#) 已于 2022-05-02 14:02:56 修改 2109 收藏 18

分类专栏: [CTF](#) 文章标签: [1024程序员节](#) [安全](#) [unctf](#) [python](#) [CTF](#)

于 2021-10-24 12:34:00 首次发布

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/Hardworking666/article/details/120932217>

版权



[CTF 专栏收录该内容](#)

21 篇文章 2 订阅

订阅专栏

文章目录

一、改高宽

二、lsb（最低有效位）隐写+base64编码图片

CRC32碰撞脚本

三、盲水印+明文攻击

傅里叶盲水印

四、IDAT块隐写

提数据+zlib解压+625二维码

png末尾藏zip

五、分离与拼接

convert分离gif+montage拼接

六、像素点合成

1、PPM格式+多种文件转换网站

七、流量类

1、wireshark提取数据流//tcpextract

tcpextract

strings

2、协议分级+导出HTTP对象

八、二维码类

1、bmp转二维码

2、16进制转pyc

pyc隐写Stegosaurus

3、二进制作二维码

4、4个值转二维码

misc文件头尾

文件隐写和图片隐写步骤

misc思路

MISC文件隐写总结(图片,音频,视频,压缩包等文件)

misc图片类总结（新赛题）

使用图片隐写的Python远控恶意样本分析

一、改高宽

打开图片发现下面好像少了什么。



WinHex打开可以看到PNG的文件头

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	PN
00000016	00	00	03	13	00	00	01	F4	08	06	00	00	00	DA	5A	4A	gAMA
00000032	50	00	00	00	01	73	52	47	42	00	AE	CE	1C	E9	00	00	PHYS
00000048	00	04	67	41	4D	41	00	00	B1	8F	0B	FC	61	05	00	00	gAMA
00000064	00	09	70	48	59	73	00	00	0E	F2	00	00	0E	F2	01	CE	PHYS
00000080	14	7B	DE	00	00	00	11	74	45	58	74	54	69	74	6C	65	{b
00000096	00	50	44	46	20	43	72	65	61	74	6F	72	41	5E	BC	28	PDF
00000112	00	00	00	13	74	45	58	74	41	75	74	68	6F	72	00	50	tEXt
00000128	44	46	20	54	6F	6F	6C	73	20	41	47	1B	CF	77	30	00	DF
00000144	00	00	2D	7A	54	58	74	44	65	73	63	72	69	70	74	69	-zTXt
00000160	6F	6E	00	00	08	99	CB	28	29	29	B0	D2	D7	2F	2F	2F	on
00000176	D7	2B	48	49	D3	2D	C9	CF	CF	29	D6	4B	CE	CF	05	00	*+HI
00000192	6E	9F	08	F1	97	AF	2C	B8	00	00	FF	30	49	44	41	54	nY
00000208	78	5E	EC	FD	C9	AF	25	CB	F2	EF	09	C5	6A	F6	DE	99	x^i
00000224	79	CE	FD	95	4A	AA	01	25	F1	54	E8	95	04	02	31	E6	yiy
00000240	1F	40	62	CA	B4	06	CC	98	52	43	84	50	4D	6A	02	13	@bE
00000256	FE	1D	46	48	48	CC	18	32	A6	A9	F7	80	7A	48	F5	EA	cd
00000272	D7	DC	7B	CF	3D	99	B9	9B	B5	D6	E6	FB	31	73	0B	B7	XU

解析:

- (固定) 八个字节89 50 4E 47 0D 0A 1A 0A为png的文件头
 - (固定) 四个字节00 00 00 0D (即为十进制的13) 代表数据块的长度为13
 - (固定) 四个字节49 48 44 52 (即为ASCII码的IHDR) 是文件头数据块的标示 (IDCH)
 - (可变) 13位数据块 (IHDR)
 - 前四个字节代表该图片的宽
 - 后四个字节代表该图片的高
 - 后五个字节依次为: Bit depth、ColorType、Compression method、Filter method、Interlace method
 - (可变) 剩余四字节为该png的CRC检验码, 由从IDCH到IHDR的十七位字节进行crc计算得到。
- 文件尾: AE 42 60 82

图片尺寸为787x500(高x宽)

- 00 00 00 0D 说明IHDR头块长为13
- 49 48 44 52 IHDR标识
- 00 00 03 13 图像的宽, 787像素
- 00 00 01 F4 图像的高, 500像素
- 发现高宽错误

这里需要注意的是, 文件宽度不能任意修改, 需要根据 IHDR 块的 CRC 值爆破得到宽度, 否则图片显示错误不能得到 flag

```

import os
import binascii
import struct

crcbp = open("D:\\桌面文件\\bingbing.png", "rb").read() #打开图片
for i in range(2000):
    for j in range(2000):
        data = crcbp[12:16] + \
            struct.pack('>i', i)+struct.pack('>i', j)+crcbp[24:29]
        crc32 = binascii.crc32(data) & 0xffffffff
        if(crc32 == 0xda5a4a50): #图片当前CRC
            print(i, j)
            print('hex:', hex(i), hex(j))

```

```

787 787
hex: 0x313 0x313

进程已结束, 退出代码为 0

```

把高宽都改成787保存得flag



二、lsb（最低有效位）隐写+base64编码图片

lsb隐写简介

攻防世界Misc高手进阶区 3-11

下载png文件，binwalk，发现zlib文件。

[binwalk详解](#)

```

root@kali:~# binwalk '/media/sf_ctf/d0430db27b8c4d3694292d9ac5a55634.png'
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0         PNG image, 1440 x 1080, 8-bit/color RGB, non-interlaced
41          0x29         Zlib compressed data, default compression

```

		Extract Preview
504b030414000008	08005db7354b64ee	PK].5Kd.
9dab83040000c807	000008000000666cf1
61672e747874c555	c992a33810fd200e	ag.txt.U ...8.. .
6215e830070c36fb	6a30cbcd0830c606	b..0..6. j0...0..
6ccc567c7ddb3ee	aaa9883974f41c46	l.V }... ...9t..F
172d914abd7c99f9	743e6c1c7f260da5	- . +> Pz

```

d489cf61efc36alb 9e9e2bf7b597a124 ...a..j. ..+....$
26cf59d2f85d875e 06626cef7d5213fb &.Y..].^ .bl.)R..
078balf73af05b2f a4364f9ba59e2721 .....: [/ .60...'!
flc2d7a18eb7558a 6751dcb64fa38d4e .....U. gQ..O..N

```

The screenshot shows a configuration window for bit plane extraction. On the left, under 'Bit Planes', there are four rows: Alpha, Red, Green, and Blue. Each row has checkboxes for bit planes 7 through 0. The 'Red', 'Green', and 'Blue' rows have the '0' checkbox checked. On the right, under 'Order settings', 'Extract By' is set to 'Row', 'Bit Order' is set to 'MSB First', and 'Bit Plane Order' is set to 'RGB'. At the bottom, there is a 'Preview Settings' section with 'Include Hex Dump In Preview' checked. A watermark 'CSDN @Hardworking666' is visible in the bottom right corner.

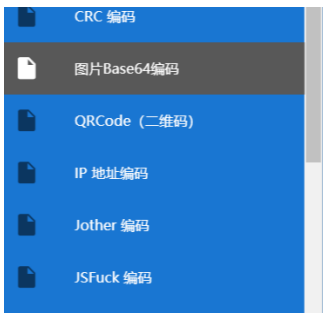
LSB隐写, save bin, 改后缀解压, 弹出已损坏, 用winrar自带的修复

The screenshot shows a file explorer window with a folder named '名称' containing a file 'flag.txt'. Below the file list, there is a 'WinRAR:诊断信息' (WinRAR: Diagnostic Information) section. It shows a warning icon and the text '压缩文件已损坏' (Compressed file is corrupted) for the file '1.zip (D:\桌面文件\1.zip)'. There are tabs for '信息' (Information) and '压缩文件' (Compressed file).

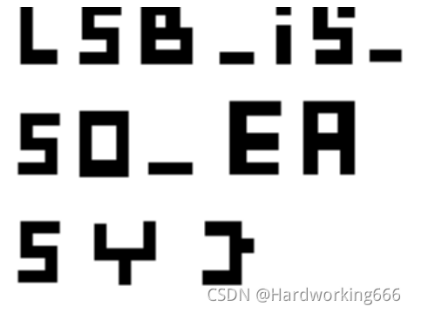
The screenshot shows a Notepad window titled 'flag.txt - 记事本'. The text content is a Base64 encoded string: `iVBORw0KGgoAAAANSUhEUgAAAPoAAAD6CAYAAAC0CFA6BAgdAgQOgT4NdWTOeoz7I46Lvsw00OA0CFA6`. The menu bar shows '文件(F)', '编辑(E)', '格式(O)', '查看(V)', and '帮助(H)'.

结尾的=号判断是base64编码, 开头iVBORw0K说明是base64编码的图片。
用captfencoder加上头, 转为图片: FLAG{LSB_i5_SO_EASY}

The screenshot shows the CaptfEncoder V2 web application. The interface is in Chinese. The main area is titled '图片Base64编码 X'. There is a text input field with the Base64 string from the previous screenshot, and a '解码' (Decode) button. Below the input field, there is a preview of the resulting image, which shows the text 'FLAG{'. The text input field has 'data:image/jpeg;base64,' added to the beginning of the string.



```
BQocLe39///zf9/xSCwSY0SFA6BAgdAgQOgQIHQKEDgGLv7329v
b2+b/1PDqVZ8dZ67uCe13HV1sc95GRcRo9x7Xe18aZ0SFA6BAgd
AgQOgR4GLdw/99d/5xz2/q61jD6Hp/pWuo2CX3NN3jrIjbuf/T6t7
6uNVzhHJnHrTsECB0ChM6fW/RHG/chdAgQOgQIHQJO8330Z6fx
7DhrfYtni7XonHPb+rpe8eoYHHGOjDGjX9AU5twNjKLnzWz9aOM
```



CRC32碰撞脚本

Misc中的有一类题目是要求我们知道加密后的rar文件中的内容。但是rar文件密码我们不知道，直接爆破密码也不是很现实。但是当文件的大小比较小，或者字符数量较少时，就可以根据crc校验码来爆破出rar内部文件的内容。

文件	大小	日期	CRC32
..			
flag_0.txt *	4	2018/9/22 23:...	7DE0AB32
flag_1.txt *	4	2018/9/22 23:...	B1441D53
flag_2.txt *	4	2018/9/22 23:...	49BD11F5
flag_3.txt *	4	2018/9/22 23:...	B42F1DFA
flag_4.txt *	4	2018/9/22 23:...	8163F43E
flag_5.txt *	4	2018/9/22 23:...	1FC8FE25

可以看到最后一列是对应文件的CRC校验码。并且每个文件只有4字节，所以可以看作每个CRC校验码都对应了唯一的文件。Python2爆破如下：

```

import binascii
import string

dic=string.printable #打印出字符表
crc1=0x7DE0AB32
crc2=0xB1441D53
crc3=0x49BD11F5
crc4=0xB42F1DFA
crc5=0x8163F43E
crc6=0x1FC8FEE5

for i in dic:
    for j in dic:
        for n in dic:
            for m in dic:
                s=i+j+n+m
                if(crc1==(binascii.crc32(s) & 0xffffffff)):
                    text1=s
                if (crc2 == (binascii.crc32(s) & 0xffffffff)):
                    text2=s
                if (crc3 == (binascii.crc32(s) & 0xffffffff)):
                    text3=s
                if (crc4 == (binascii.crc32(s) & 0xffffffff)):
                    text4=s
                if (crc5 == (binascii.crc32(s) & 0xffffffff)):
                    text5=s
                if (crc6 == (binascii.crc32(s) & 0xffffffff)):
                    text6=s
print text1+text2+text3+text4+text5+text6

```

CRC爆破的另一个脚本

三、盲水印+明文攻击

攻防世界Misc的warmup，2017ciscn（全国大学生信息安全竞赛）

下载打开，两个一样的open_forun.png，明文攻击，将open_forum.png压缩成zip，然后使用ARCHPR的明文攻击

The screenshot shows a file explorer window with the following content:

- Files in the main directory:

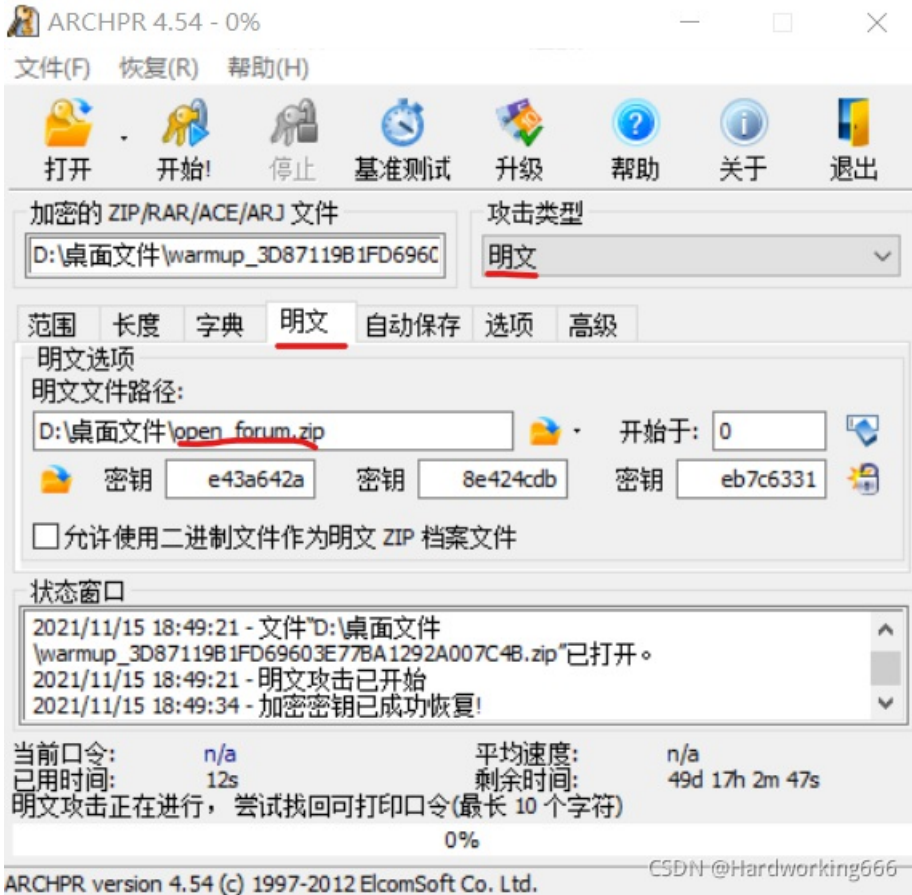
warmup_3D87119B1FD69603E77BA1292A007C4B.zip	8,406,779
open_forum.png	42,196
- Contents of the zip file (warmup_3D87119B1FD69603E77BA1292A007C4B.zip - ZIP 压缩文件):

名称	大小
fuli.png *	3,869,944
fuli2.png *	4,551,642
open_forum.png *	42,196
- Details for the selected file (open_forum.png *):

名称	大小	类型	日期	哈希
open_forum.png *	42,196	41,524 PNG 文件	2017/7/5 13:03	83E22C5E



注：两个open_forum.png的crc32的值一样，以及两个文件被压缩之后的大小，满足明文攻击要求。



解压出来是这样:



两个图，试试盲水印：

```
python bwmforpy3.py decode fuli.png fuli2.png flag.png --oldseed
```

注：如果要让python3兼容python2的random算法请加 --oldseed参数。结果就是flag.png。

```
D:\Python385\Lib\BlindWaterMark-master>python bwmforpy3.py decode fuli.png fuli2.png fuli3.png --oldseed
image<fuli.png> + image(encoded)<fuli2.png> -> watermark<fuli3.png>
D:\Python385\Lib\BlindWaterMark-master>
```



盲水印详解

傅里叶盲水印

VNCTF021 冰冰好像藏着秘密

傅里叶盲水印原理：

图片经过傅里叶变换后，水印图片直接按像素覆盖到频率域，因为频谱是中心对称的，所以加水印也要对称的加，具体就是图片分上下两部分，左上加了什么，右下也要加同样的内容。之后傅里叶反变换回去。解水印的时候变换到傅里叶变换提取就可以了。

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img = cv.imread('D:\\CTF\\FFT.png', 0) #直接读为灰度图像，不过此题已经是灰度图片了
f = np.fft.fft2(img) #做频率变换
fshift = np.fft.fftshift(f) #转移像素做幅度谱
s1 = np.log(np.abs(fshift)) #取绝对值：将复数变化成实数取对数的目的是为了将数据变化到0-255
plt.subplot(121)
plt.imshow(img, 'gray')
plt.title('original')
plt.subplot(122)
plt.imshow(s1, 'gray')
plt.title('center')
plt.show()
```

CTF盲水印详解

四、IDAT块隐写

提数据+zlib解压+625二维码

图像数据块 IDAT (image data chunk)：它存储实际的数据，在数据流中可包含多个连续顺序的图像数据块。IDAT 块只有当一个块充满时，才会继续一个新的块。

```
pngcheck.exe -v sctf.png
```

```
chunk IDAT at offset 0xf0008, length 65524
chunk IDAT at offset 0x100008, length 65524
chunk IDAT at offset 0x110008, length 65524
chunk IDAT at offset 0x120008, length 65524
chunk IDAT at offset 0x130008, length 65524
chunk IDAT at offset 0x140008, length 65524
chunk IDAT at offset 0x150008, length 45027
chunk IDAT at offset 0x15aff7, length 138
chunk IEND at offset 0x15b08d, length 0
0 errors detected in sctf.png (28 chunks, 30.8% compressed)
```

前面的块都是65524，到了0x150008变为45027，再到0x15aff7的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足的块里。

0x150008中的45027位数据是正常的图片信息。0x15aff7的138位数据是人为录入的，且所在的数据块也是人为创建的。

IDAT中的数据采用 LZ77 算法的派生算法进行压缩，所以可以用 zlib 解压缩。

可以用010 editor直接提取出数据，然后扔进zlib解压脚本里解压获得原始数据。

查看异常数据块的情况，使用010editor/winhex打开，导出异常数据块：

The screenshot shows the 010editor interface with a hex dump. Red boxes highlight the following hex values: 49 44 41 54, D9 CF A5 A8, and 78 9C 5D 91 01. A search bar at the bottom shows 'Find ASCII: IDAT'. The output window shows a list of addresses and values:

Address	Value
C0008h	IDAT
D0008h	IDAT
E0008h	IDAT

CSDN @Hardworking666

查找78 9C文件头标志，zlib。

This is a close-up of the hex dump from the previous image. The signature 78 9C 5D 91 01 is highlighted with a red box. The corresponding ASCII characters are also visible.

Hex	ASCII
95 00 FA 54 0D 21 BD BA 02 FF 3F 01 E7 98 5F 68	.úT.!!º.ÿ?.ç ^h
95 8F CD 00 00 00 8A 49 44 41 54 78 9C 5D 91 01	.Í... IDAT[ç]^

CSDN @Hardworking666

将异常的IDAT数据块斩头去尾之后使用脚本解压，在python2代码如下：

```
import zlib
import binascii
IDAT = "789C5D91011280400802BF04FFFF5C75294B5537738A21A27D1E49CFD17DB3937A92E7E603880A6D485100901FB0410153350DE8
3112EA2D51C54CE2E585B15A2FC78E8872F51C6FC1881882F93D372DEF78E665B0C36C529622A0A45588138833A170A2071DDCD18219DB8C
0D465D8B6989719645ED9C11C36AE3ABDAEFCFC0ACF023E77C17C7897667".decode('hex')
result = binascii.hexlify(zlib.decompress(IDAT))
print (result.decode('hex'))
print (len(result.decode('hex')))
```

得到压缩后的文件：

```
D:\Python2718\python.exe
1111111000100001101111111
625
```

发现是625，是一个二维码的矩阵，使用python2代码做成二维码：

```
from PIL import Image
MAX = 25
pic = Image.new("RGB", (MAX,MAX))
str = "11111110001000011011111110000010111001011010000011011101010000000010111011011101001000000001011101101110
101101101001011011000001010101101101000001111111010101010101111110000000010111011100000000110100110000010100
11101101110101010010001110000000000101000000010010011010001001110011110111001111000011101111000110010100011
00111000010101000110100011110101100000101000110111011001000011100111001000010111111010000000110101001
000111101111110111000011010110111000001000011001100011110101110100011010011111000010111010110001110100111001011
101001001110110110001100000101100011010001111110110101000110100111110000101110101100011101001110010111001011
101001001110110110001100000101100011010001100011111101101011011011011011011"
```

```
i=0
for y in range(0,MAX):
    for x in range(0,MAX):
        if(str[i] == '1'):
            pic.putpixel([x,y], (0,0,0))
        else:pic.putpixel([x,y], (255,255,255))
        i = i+1
pic.show()
pic.save("flag.png")
```

运行得到二维码

png末尾藏zip

PNG (png), 文件头: 89504E47 文件尾: 49454E44AE426082

00	00	49	45	4E	44	AE	42	60	82	03	04	14	03	01	00	IEND@B` ,
08	00	23	86	88	53	31	EF	3B	33	5E	00	00	00	B8	00	#t^S1i;3^ ,
00	00	08	00	00	00	66	6C	61	67	2E	74	78	74	49	21	flag.txtI!
83	94	AA	7E	E1	B7	AE	CD	DD	EC	3B	DE	C7	1F	99	77	f"a~á·@ÍYi;PÇ "w
CF	DB	D8	33	00	9E	A5	F9	6A	80	34	66	94	BC	4E	51	ÏÛØ3 ž¥ùje4f"4NQ
BD	F7	99	02	FC	5A	51	80	E6	5B	87	30	70	09	E2	F5	¼÷" úZQeα[+0p âð
41	8B	5E	C1	1D	6C	28	36	51	DC	C2	67	AE	CD	6E	07	A<^Á l(6QÜÂg@Ín
A3	F3	9E	38	32	98	04	37	FA	86	75	8D	4D	89	3C	23	£óž82~ 7útu M%<#
BD	5E	8E	DF	E0	B3	33	18	0E	73	DE	BA	50	4B	01	02	¼^ŽBà³3 sP°PK
3F	03	14	03	01	00	08	00	23	86	88	53	31	EF	3B	33	? #t^S1i;3
5E	00	00	00	B8	00	00	00	08	00	24	00	00	00	00	00	^ , \$
00	00	20	80	A4	81	00	00	00	00	66	6C	61	67	2E	74	€" flag.t
78	74	0A	00	20	00	00	00	00	00	01	00	18	00	00	6D	xt m
29	75	10	EC	D7	01	80	79	72	86	10	EC	D7	01	00	6D)u ix eyrt ix m
29	75	10	EC	D7	01	05	06	00	00	00	00	01	00	01	00)u ix
5A	00	00	00	84	00	00	00	00	00							CSDN @Hardworking666

从0304后全部保存，在前面和后面加上504B，后保存为zip

用stegpy得到zip密码:

```
root@kali:~# stegpy /root/桌面/cxkl.png
passw0rd@123
```

五、分离与拼接

convert分离gif+montage拼接

攻防世界MISC进阶: glance-50, mma-ctf-2nd-2016

下载拿到一个gif图片，很窄(宽2px)。

(1) 用kali的convert命令先把gif分解开：

```
convert glance.gif flag.png
```

补充：

水平镜像反转图片

```
convert -flop reverse.jpg reversed.jpg
```

垂直镜像反转图片

```
convert -flip reverse.jpg reversed.jpg
```

convert命令使用

总共分离出来200个图片。用工具：montage合成，命令：

```
montage flag*.png -tile x1 -geometry +0+0 flag.png
```

-tile是拼接时每行和每列的图片数，这里用x1，就是只一行

-geometry是首选每个图和边框尺寸，我们边框为0，图照原始尺寸即可

flag: TWCTF{Bliss by Charles O'Rear}

(2) 也可以直接用网站。GIF动态图片是由多张静态图片组合而成，按照一定的顺序和时间进行播放。该网站将GIF图片反向分解成一张张静态图。GIF图片有多少帧，就有多少张静态图片。

[GIF分解网站](#)

(3) 也可以写脚本

```
import os
from PIL import Image

im = Image.new('RGB', (2*201,600))#new(mode,size) size is Long and width
PATH = 'E:/ctf/glance.gif'

FILE_NAME = [i for i in os.listdir(PATH)]

width = 0
for i in FILE_NAME:
    im.paste(Image.open(PATH+i),(width,0,width+2,600))#box is 左,上,右,下
    width += 2
im.show()
```

六、像素点合成

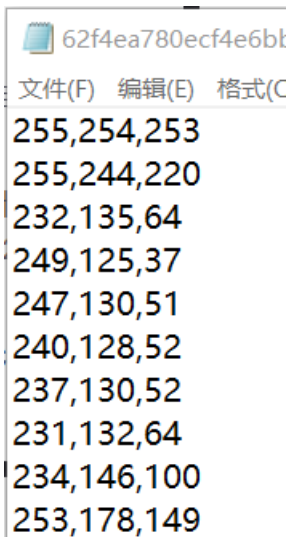
1、PPM格式+多种文件转换网站

攻防世界 Misc Miscellaneous-200 defkthon-ctf

miscellaneous

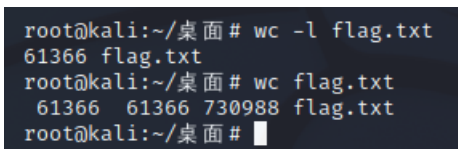
adj. 混杂的; 各种各样的;

(1) 提供的flag.txt文件每行包含由三个逗号分隔的值组成的元组。这看起来像一个给定RGB值的图像。



```
62f4ea780ecf4e6bt
文件(F) 编辑(E) 格式(C)
255,254,253
255,244,220
232,135,64
249,125,37
247,130,51
240,128,52
237,130,52
231,132,64
234,146,100
253,178,149
```

总共有61366行：



```
root@kali:~/桌面# wc -l flag.txt
61366 flag.txt
root@kali:~/桌面# wc flag.txt
61366 61366 730988 flag.txt
root@kali:~/桌面# █
```

注：Linux wc命令用于计算字数。

-l或-lines 显示行数。

-w或-words 只显示字数。

-c或-bytes或-chars 只显示Bytes数。


flag.txt文件的行数为61366、单词数61366、字节数730988

图像的尺寸是这个数字（61366）的等分，所以可能是:1,2,61,122,503,1006,30683,61366。

最可能的图像大小是 122×503px 或 503×112px 。

注：px是分辨率的单位，是英语单词pixel的缩写，意为像素(组成屏幕图像的最小独立元素)。

将文本文件转换为图像的最可以将其转换为PPM格式，其标题如下：



```
P3
122 503
255
```

注：PPM（Portable PixMap，便携式像素映射）。这些图片格式都相对比较容易处理，跟平台无关，所以称之为portable，简单理解，就是比较直接的图片格式，比如PPM，其实就是把每一个点的RGB分别保存起来。所以，PPM格式的文件是没有压缩的，相对比较大，但是由于图片格式简单，一般作为图片处理的中间文件（不会丢失文件信息），或者作为简单的图片格式保存。

PPM文件

[PPM文件格式详解](#)

然后是flag.txt的内容，逗号用空格替换。（快捷键ctrl+h 实现替换的功能）

```
*flagP3.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
P3
122 503
255
255 255 255
255 255 255
255 255 255
CSDN @Hardworking666
```

TXT到PPM转换器

这个网站可以转换许多东西！

结果是flag.ppm。用极速看图软件打开（kali中可以直接打开）：



转换为PNG，并翻转+旋转它，使它更容易阅读，结果如下图所示：

```
convert -flip -rotate 90 flag.ppm flag.png
```



(2) 分析文本发现是道画图题，直接编写 python 程序


```

from ast import literal_eval as make_tuple
from PIL import Image
f = open('flag.txt', 'r')
corl = [make_tuple(line) for line in f.readlines()]
f.close()
img0 = Image.new('RGB', (270, 270), '#ffffff')
k=0
for i in range(246):
    for j in range(246):
        img0.putpixel ([i , j], corl[k])
        k=k+1
img0.save("result.png")

```

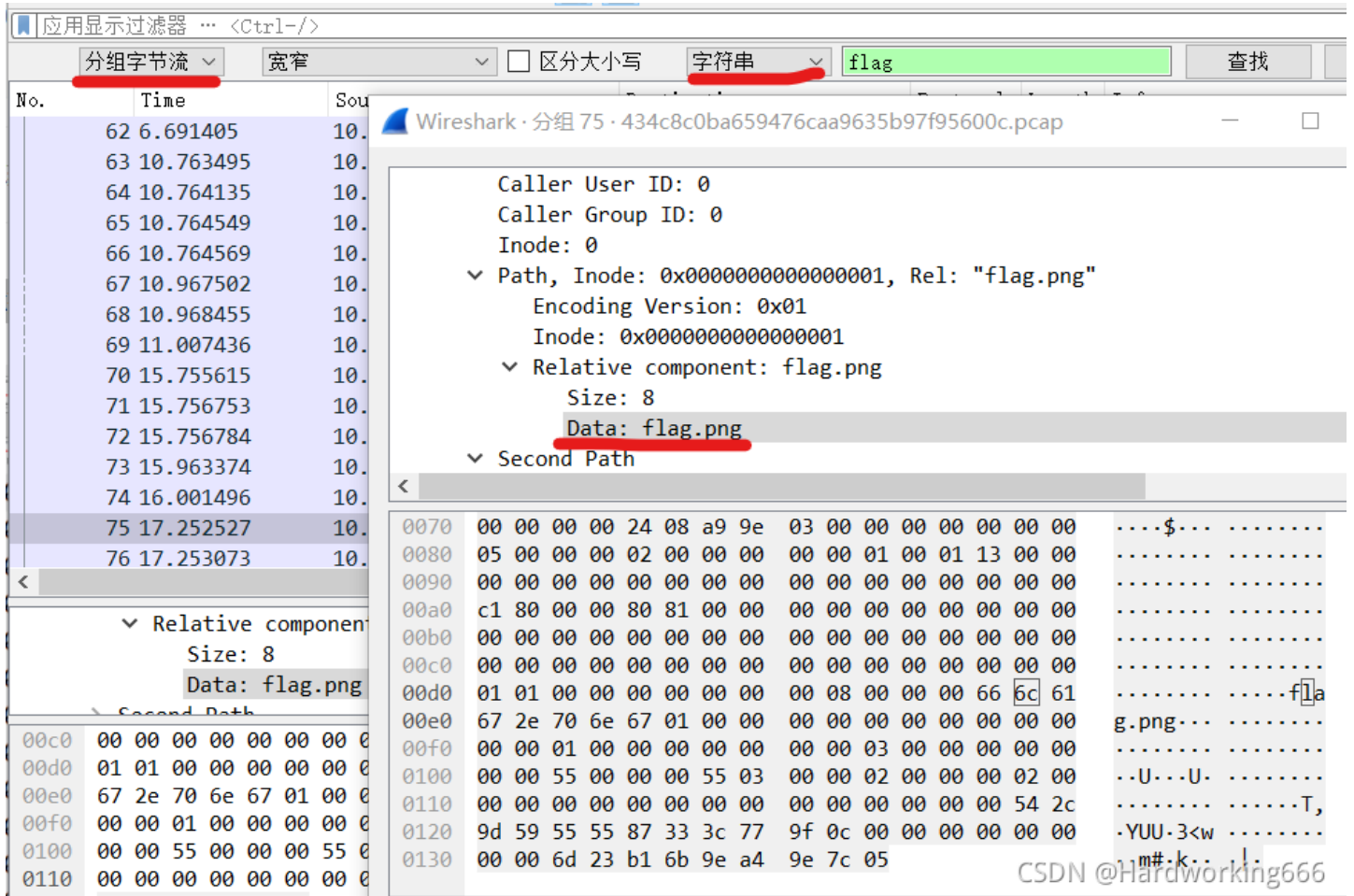
flag{ youc@n'tseeme }

七、流量类

1、wireshark提取数据流//tcpextract

攻防世界misc进阶Cephalopod

用wireshark搜索flag字符串，可以看到



PNG头的16进制为89504E47，然后搜索该16进制，找到一条TCP报文，然后追踪TCP流。

可以看到这是一个图片数据流。尾为：文件尾：AE 42 60 82。保留原始数据。



126	21.468990	10.0.2.7	10.0.2.10	TCP	2962 39618
127	21.469354	10.0.2.7	10.0.2.10	TCP	4410 39618
128	21.469528	10.0.2.7	10.0.2.10	TCP	4410 39618
129	21.469674	10.0.2.7	10.0.2.10	TCP	2962 39618
130	21.469988	10.0.2.10	10.0.2.7	TCP	66 6800 →
131	21.470125	10.0.2.7	10.0.2.10	TCP	8754 39618
132	21.470313	10.0.2.10	10.0.2.7	TCP	66 6800 →
133	21.470351	10.0.2.7	10.0.2.10	TCP	2962 39618
134	21.470403	10.0.2.7	10.0.2.10	TCP	4410 39618

Checksum: 0x2387 [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0
 > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
 > [SEQ/ACK analysis]
 > [Timestamps]
 TCP payload (2896 bytes)
[\[Reassembled PDU in frame: 308\]](#)
 TCP segment data (2896 bytes)

```
0110 00 00 00 01 00 00 00 89 50 4e 47 0d 0a 1a 0a 00 ..... PNG.....
0120 00 00 0d 49 48 44 52 00 00 06 da 00 00 09 b0 08 ... IHDR CSDN@Hardworking666
0130 06 00 00 00 1a 5f ff 77 00 00 00 06 62 4b 47 4a ..... .w ....hKGD
```

Wireshark · 追踪 TCP 流 (tcp.stream eq 2) · 434c8c0ba659476caa9635b97f95600c.pcap

129 客户端 分组, 8 服务器 分组, 8 turn(s).

整个对话 (2413kB) Show data as ASCII

查找: PNG CSDN@Hardworking666

No.	Time	Source	Destination	Protocol	Length	Info
301	21.496037	10.0.2.7	10.0.2.10	TCP	65226	39618
302	21.496071	10.0.2.7	10.0.2.10	TCP	65226	39618
303	21.496100	10.0.2.7	10.0.2.10	TCP	1514	39618

也可以直接

```
tcpextract -f 1.pcap
```

得到一张png图片 得到flag : HITB{95700d8aefdc1648b90a92f3a8460a2c}

Tcpextract是用来从网卡抓包并将其还原成文件的一个开源软件，它的基本原理是在抓取的数据包中匹配文件的特征头和特征尾。

strings

```
strings webshell.pcapng | grep {
```

strings命令在对象文件或二进制文件中查找可打印的字符串。字符串是4个或更多可打印字符的任意序列，以换行符或空字符结束。strings命令对识别随机对象文件很有用。grep命令用于查找文件里符合条件的字符串

```
strings xxx.png
```

有时可以出flag

2、协议分级+导出HTTP对象

攻防世界-互相伤害!!!

wireshark打开，协议分级，基本都是TCP流量，又以超文本传输协议为主，导出HTTP对象。

八、二维码类

1、bmp转二维码

攻防世界 Misc高手low

下来一个bmp文件，用stegsolve分析无果，但是通过观察发现是RGB的通道有问题，利用的是图片中最低位的奇偶性。

实验吧原题直接用画图另存为png格式，用StegSolve打开后，调到RGB红色位置。这里有所变化

转QR Code，即二维码（Quick Response Code）

```
# Lsb隐写
import PIL.Image as Image
img = Image.open('low.bmp')
img_tmp = img.copy()
pix = img_tmp.load()
width,height = img_tmp.size
for w in range(width):
    for h in range(height):
        if pix[w,h]&1 == 0:
            pix[w,h] = 0
        else:
            pix[w,h] = 255
img_tmp.show()
```



用QR research解码得:



2、16进制转pyc

攻防世界 MISC 适合作为桌面(世安杯)

使用stegsolve发现在绿色的低位通道中有二维码



使用二维码扫描器扫描，并将16进制数字结果在winhex中打开

已解码数据 1:

位置:(4.8,8.7)-(376.9,8.8)-(4.8,381.3)-(376.8,381.4)

颜色正常, 正像

版本: 20

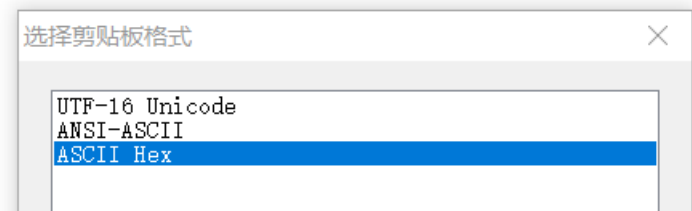
纠错等级:L, 掩码:3

内容:

```

03F30D0A79CB0558630000000000000000100000040000000730D0000006400008400
005A00006401005328020000006300000000030000001600000043000000737800000064
0100640200640300640400640500640600640700640300640800640900640A0064060064
0B00640A00640700640800640C00640D00640E00640900640F006716007D0000
6410007D0100781E007C0000445D16007D02007C01007400007C0200830100377D0100
715500577C010047486400005328110000004E6966000000696C0000006961000000696

```



注：如果错选中间这个（ANSI-ASCII），则再ASCII码转二进制（快捷键Ctrl+R）

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI	ASCII
00000000	03	F3	0D	0A	79	CB	05	58	63	00	00	00	00	00	00	00	ó	yË Xc
00000016	00	01	00	00	00	40	00	00	00	73	0D	00	00	00	64	00	@	s d
00000032	00	84	00	00	5A	00	00	64	01	00	53	28	02	00	00	00	„	z d S(
00000048	63	00	00	00	00	03	00	00	00	16	00	00	00	43	00	00	c	C
00000064	00	73	78	00	00	00	64	01	00	64	02	00	64	03	00	64	sx	d d d d
00000080	04	00	64	05	00	64	06	00	64	07	00	64	03	00	64	08	d	d d d d d
00000096	00	64	09	00	64	0A	00	64	06	00	64	0B	00	64	0A	00	d	d d d d d
00000112	64	07	00	64	08	00	64	0C	00	64	0C	00	64	0D	00	64	d	d d d d d d
00000128	0E	00	64	09	00	64	0F	00	67	16	00	7D	00	00	64	10	d	d g } d
00000144	00	7D	01	00	78	1E	00	7C	00	00	44	5D	16	00	7D	02	}	x D] }
00000160	00	7C	01	00	74	00	00	7C	02	00	83	01	00	37	7D	01		t f 7}
00000176	00	71	55	00	57	7C	01	00	47	48	64	00	00	53	28	11	qU	W GHd S(
00000192	00	00	00	4E	69	66	00	00	00	69	6C	00	00	00	69	61	Nif	il ia

00000208	00 00 00 69 67 00 00 00	69 7B 00 00 00 69 33 00	ig i{ i3
00000224	00 00 69 38 00 00 00 69	35 00 00 00 69 37 00 00	i8 i5 i7
00000240	00 69 30 00 00 00 69 32	00 00 00 69 34 00 00 00	i0 i2 i4
00000256	69 31 00 00 00 69 65 00	00 00 69 7D 00 00 00 74	i1 ie i} t
00000272	00 00 00 00 28 01 00 00	00 74 03 00 00 63 68	(t ch
00000288	72 28 03 00 00 00 74 03	00 00 00 73 74 72 74 04	r(t strt
00000304	00 00 00 66 6C 61 67 74	01 00 00 00 69 28 00 00	flagt i(
00000320	00 00 28 00 00 00 00 73	04 00 00 00 31 2E 70 79	(s 1.py
00000336	52 03 00 00 00 01 00 00	00 73 0A 00 00 00 00 01	R s
00000352	48 01 06 01 0D 01 14 01	4E 28 01 00 00 00 52 03	H N(R
00000368	00 00 00 28 00 00 00 00	28 00 00 00 00 28 00 00	(((
00000384	00 00 73 04 00 00 00 31	2E 70 79 74 08 00 00 00	s 1.pyt
00000400	3C 6D 6F 64 75 6C 65 3E	01 00 00 00 73 00 00 00	<module> s
00000416	00		

CSDN @Hardworking666

03F3开头，pyc文件。保存为.pyc然后反编译，在脚本后加上flag()，运行之后即可得到flag。

[pyc在线转换](#)

或者，使用“uncompyle6 文件路径\文件名.pyc > 文件路径\文件名.py”命令

```
D:\Python385\Lib\site-packages\uncompyle6\bin
```

pyc隐写 Stegosaurus

```
cy@kali:~/stegosaurus$ python3 stegosaurus.py -x 1.pyc
Extracted payload: Flag{HiD3_Pal0ad_1n_Python}
```

Stegosaurus 是一款隐写工具，它允许我们在 Python 字节码文件(pyc 或 pyo)中嵌入任意 Payload 。由于编码密度较低，因此我们嵌入 Payload 的过程既不会改变源代码的运行行为，也不会改变源文件的文件大小。Payload 代码会被分散嵌入到字节码之中，所以类似 strings 这样的代码工具无法查找到实际的 Payload 。Python 的 dis 模块会返回源文件的字节码，然后我们就可以使用 Stegosaurus 来嵌入 Payload 了。

pyc文件存在无效空间，修改后大小不变，不影响运行，可以隐藏信息。

stegosaurus.py可以隐藏和解密pyc文件中隐藏的信息。

只能利用pyc文件中的无效空间(Python3.6后参数会占1字节，如果没有参数用0x00填充，运行时被忽略，修改不影响)，编码密度较低，嵌入 Payload 后不会改变源代码的正常运行和大小，不容易被发现。

同时 Payload 会被分散嵌入到字节码之中，类似 strings 这样的代码工具无法查找到实际的 Payload 。

通过github下载后(<https://github.com/AngelKitty/stegosaurus>)运行 python stegosaurus.py -h

```
C:\Users\z' \Documents\CTF\py\stegosaurus-master>python stegosaurus.py -h
usage: stegosaurus.py [-h] [-p PAYLOAD] [-r] [-s] [-v] [-x] [-e EXPLODE]
                    carrier

positional arguments:
  carrier                Carrier py, pyc or pyo file

optional arguments:
  -h, --help            show this help message and exit
  -p PAYLOAD, --payload PAYLOAD
                        Embed payload in carrier file
  -r, --report          Report max available payload size carrier supports
  -s, --side-by-side    Do not overwrite carrier file, install side by side
                        instead.
  -v, --verbose         Increase verbosity once per use
  -x, --extract         Extract payload from carrier file
  -e EXPLODE, --explode EXPLODE
                        Explode payload into groups of a limited length if
                        necessary
                                                                CSDN @Hardworking666
```

可以发现有很多参数，-p 要隐藏的文本，-r 显示最大隐藏字节，-x可以解密。

stegosaurus解密

使用 python stegosaurus.py py_py_py.pyc -x 得到如下结果 Extracted payload: Flag{HiD3_Pal0ad_1n_Python} 。

[Stegosaurus详解](#)

3、二进制制作二维码

攻防世界 Misc很普通的数独

下载发现是一堆数独图片，把有数字的记为1，没有数字的记为0，结果保存在txt文本中。也可以调节文件位置后用画图拼接，并将有数字的格涂黑。


```
# -*- coding:utf-8 -*-
from PIL import Image
x = 45
y = 45

im = Image.new("RGB", (x, y)) # 创建图片
file = open('1.txt', 'r') # 打开rbg值文件
for i in range(0, x):
    line = file.readline() # 获取一行
    for j in range(0, y):
        if line[j] == '0':
            im.putpixel((i, j), (255, 255, 255)) # rgb转化为像素
        else:
            im.putpixel((i, j), (0, 0, 0)) # rgb转化为像素
im.show()
```

扫描得到一串字符串，base64多次解码得到flag: flag{y0ud1any1s1}

4、4个值转二维码

2019西湖论剑网络安全技能大赛（大学生组）-奇怪的TTL字段

发现ttl.txt中的ttl只有4个值63,127,191,255，写出他们的二进制表示后发现只有最高两位不同（高两位比特的数在数据传输中不容易受影响），拿下来，每4个TTL值凑出一个字节的二进制数来

63=00111111

127=01111111

191=10111111

255=11111111

于是考虑做如下转换，发现写出来的16进制数开头是ffd8，应该是jpg，于是写入文件中：

```
fp = open('ttl.txt','r')
a = fp.readlines()
p = []
for i in a:
    p.append(int(i[4:]))
s = ''
for i in p:
    if i == 63:
        a = '00'
    elif i == 127:
        a = '01'
    elif i == 191:
        a = '10'
    elif i == 255:
        a = '11'
    s += a
# print(s)

import binascii
flag = ''
for i in range(0,len(s),8):
    flag += chr(int(s[i:i+8],2))
flag = binascii.unhexlify(flag)
wp = open('res.jpg','wb')
wp.write(flag)
wp.close()
```

写完之后发现只有二维码的一部分，应该是不止一张图，用foremost直接分开就好了，之后用PPT拼在一块，扫描之后得到如下信息：

key:AutomaticKey cipher:fftu{2028mb39927wn1f96o6e12z03j58002p}

是AutoKey加密，用在线网站解密得flag

autokey解密

自动密钥密码（Autokey Cipher）也是多表替换密码，与维吉尼亚密码类似，但使用不同的方法生成密钥。