CTF misc图片类总结(入门级)





CTF 专栏收录该内容

文章目录

订阅专栏

21 篇文章 2 订阅

- 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//tcpxtract

tcpxtract

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

bingbing.png																	
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	HPNG IHDR
00000016	00	00	03	13	00	00	01	F4	08	06	00	00	00	DA	5A	4A	ô ÚZJ
00000032	50	00	00	00	01	73	52	47	42	00	AE	CE	1C	E9	00	00	P sRGB &Î é
00000048	00	04	67	41	4D	41	00	00	B1	8F	0B	FC	61	05	00	00	gAMA ± üa
00000064	00	09	70	48	59	73	00	00	0E	F2	00	00	0E	F2	01	CE	pHYs ò ò Î
08000000	14	7B	DE	00	00	00	11	74	45	58	74	54	69	74	6C	65	{ tEXtTitle
00000096	00	50	44	46	20	43	72	65	61	74	6F	72	41	5E	BC	28	PDF CreatorA^4(
00000112	00	00	00	13	74	45	58	74	41	75	74	68	6F	72	00	50	tEXtAuthor P
00000128	44	46	20	54	6F	6F	6C	73	20	41	47	1B	CF	77	30	00	DF Tools AG Iw0
00000144	00	00	2D	7A	54	58	74	44	65	73	63	72	69	70	74	69	-zTXtDescripti
00000160	6F	6E	00	00	08	99	CB	28	29	29	B0	D2	D7	2F	2F	2F	on ™Ë())°Ò×///
00000176	D7	2B	48	49	D3	2D	C9	CF	CF	29	D6	4B	CE	CF	05	00	×+HIÓ-ÉII)ÖKÎI
00000192	6E	9F	08	F1	97	AF	2C	B8	00	00	FF	30	49	44	41	54	nY n-, YOIDAT
00000208	78	5E	EC	FD	C9	AF	25	CB	F2	EF	09	C5	6A	F6	DE	99	x^iýÉ %Éði Åjö⊅™
00000224	79	CE	FD	95	4A	AA	01	25	F1	54	E8	95	04	02	31	E6	yîý•J* %ñTè• 1æ
00000240	1F	40	62	CA	B4	06	CC	98	52	43	84	50	4D	6A	02	13	@bÊ' Ì RC, PMj
00000256	FE	1D	46	48	48	CC	18	32	A6	A9	F7	80	7A	48	F5	EA,	B. EHHA 1219 +€ZHOA
00000272	D7	DC	7B	CF	3D	99	B9	9B	B5	D6	E6	FB	31	73	0B	B7-	TYDE TAL STORATS
	-	-		-	-			-	0.00	-	-		-		-		and a of shares o

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ASCI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() # <i>打开图片</i>
for i in range(2000):
for j in range(2000):
<pre>data = crcbp[12:16] + \</pre>
<pre>struct.pack('>i', i)+struct.pack('>i', j)+crcbp[24:29]</pre>
<pre>crc32 = binascii.crc32(data) & 0xffffffff</pre>
if(crc32 == 0xda5a4a50): # <i>图片当前CRC</i>
print(i, j)
<pre>print('hex:', hex(i), hex(j))</pre>

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 laced	0×0	PNG image, 1440 x 1080, 8-bit/color RGB, non-int
41	0x29	Zlib compressed data, default compression

			Extract	Preview
504b030414000008	08005db7354b64ee	PK.].5Kd.
9dab83040000c807	000008000000666c			fl
61672e747874c555	c992a33810fd200e	ag.t	xt.U	8
6215e830070c36fb	6a30cbcd0830c606	b0)6.	j00
6ccc567c7ddbd3ee	aaa9883974f41c46	1.V	}	9tF
1724014=bd7c00f0	743=6c1c7f260d=5		т	+∧1 ∏£

 d489cf6lefc36alb 9e9e2bf7b597al24
 ...a..j. ..+...\$

 26cf59d2f85d875e 06626cef7d52l3fb
 &.Y..].^ .bl.}R..

 078balf73af05b2f a4364f9ba59e272l
[/ .60...'!

 flc2d7al8eb7558a 675ldcb64fa38d4e
U. gQ..0.N

Rit Planes	Order settings
Alpha 7 6 5 4 3 2 1 0	Extract By Row Column
Red 7 6 5 4 3 2 1 1 0	Bit Order 💿 MSB First 🔾 LSB First
Green 7 6 5 4 3 2 1 2 0	Bit Plane Order
Blue 7 6 5 4 3 2 1 2 0	● RGB ○ GRB
	ORBG OBRG
Preview Settings Include Hex Dump In Preview	○ GBR ○ BGR CSDN @Hardworking666

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



🥘 flag.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

iVBORw0KGgoAAAANSUhEUgAAAPoAAAD6CAYAAAC 0CFA6BAgdAgQOgT4NdWTOeoz7I46Lvswo0OA0CFA6

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}

=	CaptfEncoder V2	2	捜索
•	编码转换	~	首页 图片Base64编码×
	古典加密	~	图片Base64编码 1 ● +
	密码学	~	编码 解码 解码
	公钥密码	~	
	其他编码	^	
	Brainfuck 编码		AD6CAYAAACI7Fo9AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjw v8YQUAAAAJcEhZcwAAEnQAABJ0Ad5mH3gAAAVqSURBVHhe7
	Bubble babble 编码		d1bTuRGAEDRiftfK1tlxgNRJNINbpff9xzJmvlpP6q5KpeB5u3v334 Bt/bX57/AjQkdAoQOAUKHAKFDgNAhQOgQIHQIEDoECB0ChA4



BQocLe39///zf9/xSCwSY0SFA6BAgdAgQOgQlHQKEDgGLv7329v b2+b/1PDqVZ8dZ67uCe13HV1sc95GRcRo9x7Xel8aZ0SFA6BAgd AgQOgR4GLdw/99d/5xz2/q61jD6Hp/pWuo2CX3NN3jrlJbuf/T6t7 6uNVzhHJnHrTsECB0ChM6fW/RHG/chdAgQOgQlHQJ08330Z6fx 7DhrfYtni7XonHPb+rpe8eoYHHG0jDGjX9AU5twNJkLnzwz9aOM



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。

"		文件夹	
flag_0.txt *	4	16 文本文档	2018/9/22 23: 7DE0AB32
flag_1.txt *	4	16 文本文档	2018/9/22 23: B1441D53
flag_2.txt *	4	16 文本文档	2018/9/22 23: 49BD11F5
flag_3.txt *	4	16 文本文档	2018/9/22 23: B42F1DFA
flag_4.txt *	4	16 文本文档	2018/9/22 23: 8163F43E
flag_5.txt *	4	16 文本文档	2018/9722 25 1ard 19 26 19 26 56

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 # <i>打印出字符表</i>
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
<pre>if(crc1==(binascii.crc32(s) & 0xffffffff)):</pre>
text1=s
<pre>if (crc2 == (binascii.crc32(s) & 0xffffffff)):</pre>
text2=s
<pre>if (crc3 == (binascii.crc32(s) & 0xffffffff)):</pre>
text3=s
<pre>if (crc4 == (binascii.crc32(s) & 0xffffffff)):</pre>
text4=s
<pre>if (crc5 == (binascii.crc32(s) & 0xffffffff)):</pre>
text5=s
<pre>if (crc6 == (binascii.crc32(s) & 0xffffffff)):</pre>
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

I. Image: Second Science of Content of Conten	8,406,779 42,196	
↑ ▲ warmup_3D87119B1FD69603E77BA1292A007	C4B.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	
open_forum.png *	42,196 41,524 PNG 文件	2017/7/5 13:03 83E22C5E
🔯 open_forum.zip		
文件(F) 命令(C) 工具(S) 收藏夹(O) 选项(N) 帮助(H)		

添加	「 解 圧到	<u>〕</u> 测试	查看	删除	全 我	向导	信息	いた 扫描病毒	注释	▲▲▲▲▲ 自解压格式		
1	open_for	um.zip -	ZIP 压缩文	件, 解包大	、小为 42,1	96 字节						
名称							大小	压缩后大小	类型		修改时间	CRC32
1.									文件	夹		
e open	forum.png	I					42,196	41,512	PNG	文件	:52077992/319244.0	063623656

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。

🔏 ARCHPR 4.54 - 0%		8 <u>6 - 1</u> 8		
文件(F) 恢复(R) 帮助(H)				
🗳 . 移 🙈 🤇	3 🤣	2	1	
打开 开始! 停止 基准	测试 升级	帮助	关于 退出	
加密的 ZIP/RAR/ACE/ARJ 文件	攻击类型	<u>u</u>		
D:\桌面文件\warmup_3D87119B1FD6	5960 明文		`	1
范围 长度 字典 明文 自述 明文选项 明文文件路径:	动保存 选项 ;	高級		
D:\桌面文件\open_forum.zip		・ 开始于:	D 🔊	
📄 密钥 e43a642a 密钥	8e424cdb	密钥	eb7c6331 🔒	
□ 允许使用二进制文件作为明文 Z	IP 档案文件			
状态窗口				_
2021/11/15 18:49:21 - 文件"D:\桌面)	文件 2924007C48 zin"	⊐+TII.	,	^
2021/11/15 18:49:21 - 明文攻击已开	292A007C-10.21D [始	21174 0		1
2021/11/15 18:49:34 - 加密密钥已成	功恢复!			~
当前口令: n/a 已用时间: 12s 明文攻击正在进行, 尝试找回可打印	平均速度; 剩余时间; 1口令 <mark>(</mark> 最长 10 个?	: n/a : 49d 1 字符)	7h 2m 47s	
	0%	CSDALG	i Llovelovereleiner 66	6
ARCHPR version 4.54 (c) 1997-2012 Elco	mSoft Co. Ltd.	CSDING	ertal divorkingeo	0

加密密钥已成功恢复!		\times
Advanced Archive Pa	ssword Recovery 统计信息:	
总计口令	n/a	
总计时间	12s 787ms	
平均速度(口令/秒)	n/a	
这个文件的口令	未找到	4
加密密钥	[e43a642a 8e424cdb eb7c6331]	6
	✔c确定@Hardworking	666

解压出来是这样:





fuli2.png

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>

n flagibWm_Are_W0nderfu1

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。



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 块只有当上 一个块充满时,才会继续一个新的块。

chunk	IDAT	at	offset	0xf0008,	1ength 65524	
chunk	IDAT	at	offset	0x100008,	1ength 65524	
chunk	IDAT	at	offset	0x110008,	1ength 65524	
chunk	IDAT	at	offset	0x120008,	1ength 65524	
chunk	IDAT	at	offset	0x130008,	1ength 65524	
chunk	IDAT	at	offset	0x140008,	1ength 65524	
chunk	IDAT	at	offset	0x150008,	1ength 45027	
chunk	IDAT	at	offset	0x15aff7,	length 138	
chunk	IEND	at	offset	0x15b08d,	1ength 0	
o erroi	rs det	tect	ted in s	sctf.phg t(28 chunks, 30.8% Icompres	Þ

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。



查找789C文件头标志, zlib。

02 FF 3F 01 F7 98 5F 68 1.úT.!½º.ÿ?.¢I^h 95 OO FA 54 OD 21 BD BA 95 8F CD 00 00 00 8A 49 44 41 54 78 9C 5D 91 01 I.Í...IIDA<mark>rylı</mark>'. .∎@..¿.ÿÿ∖u)KU7s 12 80 40 08 02 BF 04 FF FF 5C 75 29 4B 55 37 73 8A 21 A2 7D 1E 49 CF D1 7D B3 93 7A 92 E7 E6 03 |!¢}.IÏÑ}³|z′çæ. 88 OA 6D 48 51 00 90 1F BO 41 01 53 35 0D E8 31 Ⅰ.mHQ...°A.S5.è1 12 EA 2D 51 C5 4C E2 E5 85 B1 5A 2F C7 8E 88 72 .ê-QÅLâå|±Z∕Ç∣∣r F5 1C 6F C1 88 18 82 F9 3D 37 2D EF 78 E6 65 B0 õ.oÁ∣.∣ù=7-ïxæe° ÃlR∎" ¤U∎.∎3ip¢. C3 6C 52 96 22 A0 A4 55 88 13 88 33 A1 70 A2 07 1D DC D1 82 19 DB 8C OD 46 5D 8B 69 89 71 96 45 .ÜÑ∎.Û∎.F]∎i∎q∎E ED 9C 11 C3 6A E3 AB DA EF CF CO AC FO 23 E7 7C íⅠ.Ãjã≪ÚïÏÀ¬ð#ç| 17 C7 89 76 67 D9 CF A5 A8 00 00 00 00 49 45 4E .Ç∣vgÙÏ¥¨....IEN 44 AE 42 60 82 CSDMB 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
<pre>pic = Image.new("RGB",(MAX,MAX))</pre>
str ="11111110001000011011111111000001011100101
101110110100101110110000010101011011010000
11101101111010100001000001110000000000
00111000010101000110100011110101100000101
0001111011111101110000110101101110000010000
10100100111011011000110000101100011010001111
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

1	Ŀт	FA	E E	EA	r A	12	10	50	52	00	DC	11	гБ	10	00	00	auyeu A- mo uA
2	00	00	49	45	4E	44	AE	42	60	82	03	04	14	03	01	00	IEND@B`,
}	80	00	23	86	88	53	31	EF	3B	33	5E	00	00	00	B8	00	#†^S1ï;3^ ,
Ł	00	00	08	00	00	00	66	6C	61	67	2E	74	78	74	49	21	flag.txtI!
)	83	94	AA	7E	E1	в7	AE	CD	DD	EC	3B	DE	C7	1F	99	77	f″ª~á °GÍÝì;ÞÇ ™w
5	CF	DB	D8	33	00	9E	A5	F9	6A	80	34	66	94	BC	4E	51	ÏÛØ3 ž¥ùj€4f″≒NQ
2	BD	F7	99	02	FC	5A	51	80	E6	5B	87	30	70	09	E2	F5	½÷™ üZQ€æ[‡0p âõ
3	41	8B	5E	C1	1D	6C	28	36	51	DC	C2	67	AE	CD	6E	07	A<^Á l(6QÜÂg@Ín
Ł	A3	F3	9E	38	32	98	04	37	FA	86	75	8D	4D	89	3C	23	£óž82~ 7ú†u M‰<#
)	BD	5E	8E	DF	E0	в3	33	18	0E	73	DE	BA	50	4B	01	02	½^Žßà°3 s₽°PK
5	3F	03	14	03	01	00	08	00	23	86	88	53	31	EF	3B	33	? #†^S1ï;3
2	5E	00	00	00	B8	00	00	00	08	00	24	00	00	00	00	00	^ , ş
3	00	00	20	80	A4	81	00	00	00	00	66	6C	61	67	2E	74	€¤ flag.t
Ł	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 ì× €yr† ì× m
5	29	75	10	EC	D7	01	05	06	00	00	00	00	01	00	01	00)u ì×
2	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

C	HEX	WinHe	ex - [t	2.zip	0]																			
4	HEX	文件(F)	编辑	咠(E)	搜	索(S)	导	航(N	N) 置	至看(V)	工具(T) ₹	₹ <u>₩</u>	_具(I) 进	项(0) E	窗口()	W) 帮助(H)			
· uj			20	r		1		•	C 4	101 010		A #	HEX	¢⊊ A	ix	-		⊢ →		330	— <i>></i>	1	\$	85
	cxk	l.png	t2.zip																					
	0	offse	t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		ANSI	ASC	CII	
1	0.0	00000	00	50	4B	03	04	14	03	01	00	08	00	23	86	88	53	31	EF	PK	1	# + ^ s	51ï	
l	00	00000	16	3B	33	5E	00	00	00	B 8	00	00	00	08	00	00	00	66	6C	;3^	3		fl	
l	00	00000	32	61	67	2E	74	78	74	49	21	83	94	AA	7E	E1	в7	AE	CD	ag.txt	I!f"	ª~á	·©Í	
l	00	00000	48	DD	EC	3B	DE	C7	1F	99	77	CF	DB	D8	33	00	9E	A5	F9	Ýì;ÞÇ	™wÏÛ¢	Ø3 3	ž¥ù	
l	00	00000	64	6A	80	34	66	94	BC	4E	51	BD	F7	99	02	FC	5A	51	80	j€4f"	NQ	* ü2	ZQ€	
l	00	00000	80	E6	5B	87	30	70	09	E2	F5	41	8B	5E	C1	1D	6C	28	36	æ[‡0p	âõA< '	À I	L (6	
l	00	00000	96	51	DC	C2	67	AE	CD	6E	07	A3	F3	9E	38	32	98	04	37	QÜÂg©Í	n £ó	ź821	7	
l	00	00001	12	FA	86	75	8D	4D	89	3C	23	BD	5E	8E	DF	E0	B3	33	18	útu Ma	<#12^2	Žßà	3	
,	00	00001	28	0E	73	DE	BA	50	4B	01	02	3F	03	14	03	01	00	08	00	sþ°PK	2			
b	00	00001	44	23	86	88	53	31	EF	3B	33	5E	00	00	00	B 8	00	00	00	#†^S1ï	;3^	3		
5	00	00001	60	08	00	24	00	00	00	00	00	00	00	20	80	A4	81	00	00	Ş		€¤		
7	00	00001	76	00	00	66	6C	61	67	2E	74	78	74	0A	00	20	00	00	00	flag	.txt			
i.	0.0	00001	92	00	00	01	00	18	00	00	6D	29	75	10	EC	D7	01	80	79		m)u	ì×	€у	
3	00	00002	80	72	86	10	EC	D7	01	00	6D	29	75	10	EC	D7	01	05	06	rt ì×	m)u	ì×		
5	00	00002	24	00	00	00	00	01	00	01	00	5A	00	00	00	84	00	00	00		Z	"		
21 0	00	00002	40	00	00	50	4B	00	00	00	00	00								CSDN @	Hardv	vork	ing6	66

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。

<u> </u> 62f4e	a780eo	f4e6bk
文件(F) 练	扁辑(E)	格式(C
255,254	,253	
255,244	,220	
232,135	64,	
249,125	5,37	
247,130),51	
240,128	3,52	
237,130),52	
231,132	2,64	
234,146	6,100	
253,178	3,149	

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

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), '#fffffff')
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//tcpxtract

攻防世界misc进阶Cephalopod 用wireshark搜索flag字符串,可以看到

回し	用显示过滤器 … <ctrl-></ctrl->																
	分组字节流 🗸 🛛 宽窄			~ 🗌	区分大小	۱. ۱	字符	串	×	fl	ag					查找	
No.	Time	Sou	-			- ··					-				1 - 0		
	62 6.691405	10. 🦲	Wires	nark • 5	}组75·	434c8	3c0ba	6594	476ca	a963	35b97	9560	0c.p	сар		_	
	63 10.763495	10.															
	64 10.764135	10.		Cal]	ler Use	er ID	: 0										
	65 10.764549	10.		Cal]	ler Gro	oup Il	D: 0										
	66 10.764569	10.		Inoc	le: 0												
	67 10.967502	10.	`	∕ Path	n, Inoc	le: 0	x0000	000	0000	0000	1, Re	1: "	flag	.png	"		
	68 10.968455	10.		E	ncodin	g Ver	rsion	: 0	x01								
	69 11.007436	10.		I	node:	0x000	00000	000	00000	91							
	70 15.755615	10.		~ R	elativ	e con	npone	nt:	flag	g.pn	g						
	71 15.756753	10.			Size	: 8											
	72 15.756784	10.			Data	: fla	ng.pn	g									
	73 15.963374	10.	``	 Second 	ond Pat	th											
	74 16.001496	10.	<														
	75 17.252527	10.	0070	00 00	00 00	24 0	8 a9	9e	03	00 O	00 00	00 0	0 00	00 0	••••\$•••		· · ·
	76 17.253073	10.	0080	05 00	00 00	02 0	0 00	00	00	00 0	00 10	01 1	.3 00	9 00			• • •
<			0090	00 00	00 00	00 0	0 00	00	00	00 0	00 00	00 0	0 00	00			• • •
	✓ Relative compo	nen	00a0	c1 80	00 00	80 8	1 00	00	00	00 0	00 00	00 0	0 00	00 6			• • •
	Size: 8		0000	00 00	00 00	00 0	0 00	00	00	00 0	00 00	00 0	0 0	00 0			
	Data: flag.	png	00C0	00 00 01 01	00 00	00 0	0 00	00 00	00	00 0 02 0	00 00	00 0	6 6	- 61			fila
	Cocond Dath		0000 00e0	67 2e	70 6e	67 0	1 00	00	60	00 0 00 0	0 00	00 0		01	g nng		тца
00c0	00 00 00 00 00 00 00	90 0	00f0	00 00	01 00	00 0	0 00	00	00	00 0	3 00	00 0	0 0	00			
00d0	01 01 00 00 00 00 0	90 0	0100	00 00	55 00	00 0	0 55	03	00	00 0	2 00	00 0	0 0	2 00	··U···U·		
00e0	67 2e 70 6e 67 01 0	90 0	0110	00 00	00 00	00 0	0 00	00	00	00 e	00 00	00 0	0 54	4 2c			٠т,
00100	00 00 01 00 00 00 0	00 0	0120	9d 59	55 55	87 3	3 3c	77	9f	0c 0	00 00	00 0	0 00	00 0	•YUU•3 <v< th=""><th>۰</th><th></th></v<>	۰	
0100	00 00 55 00 00 00 00		0130	00 00	6d 23	b1 6	b 9e	a4	9e	7c 0)5		C	SDN	เ⊚ <mark>∺#</mark> #:k.:	orking	666
0110		90.6												551	Ginnaw	21/11/2/	500

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。

凤应月	围显示ì	过滤器 …	<ctrl-></ctrl->								
	分组。	字节流 ~	宽窄		🔷 🗌 区分大小	写	十六进制值 ~	89504	347		
No.		Time		Source		Desti	ination		Protocol	Length	Inf∘
	400	24 4600	00	40 0 0 7		10.0	2 40		TOD	2062	20640

	126 21.468990	10.0.2./	10.0.2.10	ICP	2962 39618
	127 21.469354	10.0.2.7	10.0.2.10	ТСР	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	ТСР	2962 39618
	134 21.470403	10.0.2.7	10.0.2.10	ТСР	4410 39618
<					
	Checksum: 0x2387	[unverified]			
	[Checksum Status	: Unverified]			
	Urgent Pointer:	0			
	> Options: (12 byt	es), No-Operation	n (NOP), No-Operation (NOP)	, Timestamps	
	> [SEQ/ACK analysi	s]			
	> [Timestamps]				
	TCP payload (289	6 bytes)			
	[Reassembled PDU	in frame: 308]			
	TCP segment data	(2896 bytes)			
01:	10 00 00 00 01 00	00 00 <mark>89 50 4e</mark> 4	47 0d 0a 1a 0a 00 ······	• PNG	
01	20 00 00 0d 49 48	44 52 00 00 06 0	da 00 00 09 b0 08 ···IHDR	 CSDN @Hard 	working666
01	20 06 00 00 00 11	5f ff 77 00 00 0	00 06 62 Ab A7 AA	w hKGD	

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



📘 tcp.	.strea	am eq 2								
	分组织	字节流 ~	宽窄		🔷 🗌 区分大小	写 十六进制值	× AE 42	60 82		
No.		Time		Source		Destination		Protocol	Length	Inf∘
	301	21.496037		10.0.2.7		10.0.2.10		ТСР	65226	39618 -
+	302	21.496071		10.0.2.7		10.0.2.10		ТСР	65226	39618 -
	202	21 496100		10027		10 0 2 10		тср	1514	39618 -

ΙT			CI.4.	1010	0		11		2.1						то.	0.2	. 10		10		171-	- 55	010
		304 3	21.49	9619	8		10	0.0.	2.10)					10.	0.2	.7		TC	Р	66	68	00 →
•		305	21.49	9621	.0		10	0.0.	2.7						10.	0.2	.10		TC	Р	20338	39	618 -
		306	21.49	9660	7		10	0.0.	2.10)					10.	0.2	.7		TC	Р	66	68	00 →
+		307 3	21.49	9662	2		10	0.0.	2.7						10.	0.2	.10		TC	Р	43506	5 39	618 -
		308	21.49	9664	7		10	0.0.	2.7						10.	0.2	.10		Ce	ph	48463	0S	D Ope
		309 3	21.49	9700	1		10	0.0.	2.10)					10.	0.2	.7		TC	Р	66	68	00 →
<																							
			Kin	d: T	ime	St	amp	Opt	tion	(8)												
			Len	gth:	10																		
			Tim	esta	mp	val	ue:	248	391														
			Tim	esta	mp	ech	o r	eply	/: 2	723	68												
	> [SEQ/	АСК	ana	lysi	is]																	
	> [Time	stam	ps]																			
	T	CP p	aylo	ad	(483	397	byt	tes)															
	Т	CP s	egme	nt (data	a (4	4839	97 b	ytes	5)													
1	[100	Dor		hl.	л т	ึกก	۰	mont	- ('	о́л1-	150	o hi	/t-0	-۱۰	#1	<u> </u>	000E1 #107	(15/	1/1 +	+1 20/1	12111 H	1 20	10001
<																							
bo	e0	0 9 9	6 99	8c	89	0b	2e	e3	c8	52	42	54	ed	22	81	dc	· · · · · · · · ·	- RB	3T · " · ·				
bo	f0	5e 1	.d 72	46	dd	b3	9d	02	12	b2	35	39	9b	b0	b2	18	^·rF····	5	<u>.</u>				
bo	00	78 Ł	6 da	0f	17	a5	ef	42	fa	7e	89	8d	ca	29	cf	77	x · · · · · B	- ~ -)-w	I			
bo	10	cb f	b 5b	4c	d8	a5	06	7b	72	00	39	f1	b7	eb	13	71	··[L···{	r.9)q	1			
bo	120	a8 9	d 12	36	ta	03	4/	e1	21	93	c5	46	22	92	00	00	···6··G·	/	+"				
bo	130	00 0	0 49	45	4e	44	ae	42	- 60 E C	82	†/	94	20	23	00	00	- · TEND • B	VC	SDN (@Har	dworki	ngƙ	66
DC	140	00 0	0 56	15	4a	a۷	Øđ	oe	30	83	00	C2	ZT	8C	60		··P·J··n	V	1 11914	Sundi	GWOINI	150	00

47079e0cfffffff140000003130303030303030303030312e303030303030303030010001220000000000
ffffffff0100000023cb2400fefffffffffffffffff010000000000000000
0009b00806000000145fff7700000006624b474400ff00ff00ff00ffa0bda793000000097048597300002e2300002e230178a53f760000000774494d4
507e1081509032920357ecb0000001974455874436f6d6d656e74004372656174656420776974682047494d5057810e17000020004944415478da
ecbd5976ecca8e24ead000b2de9bff60511f5b21910c926e66303843274b6b659ebba508d21bf48d2172fc4f8e31c618f1ef3fe3fb9fbb9fb8f8b
dfa39f5e7f5fced7b8eef3cfbcc30ac5d7d2efa8eab7dccbe7bf67bc73d7cca333acf7846f74fae55f9de1d8d3ae877d53d1fd78af042c7fe54de
62e5e56a3eb9e38371c10f4799dbcd33d57376ca7f54375475d7d33fd5f57cda7e3e711f4f9e116a5374ee53796797cda5ca89f147f9f669fe3c9
edd2a7f81fd6cd7ba587d8ff2906bbd5dbc79b7de8a9ff369bec42ad98bcaa08aec73df03faae4fb9ffa76587720e085dccd6ead26f55bd8bacf3
13743022d33f79fd4e19a8c8b84fb4b52bb2ba83ff2b34366ee4c818ebed862ef9f8d41a3ec11798adc3e903adf421ffb7c51f56e95e56beffc53
855576c9dfdeebfdf7dfdfe220d879927448310d6d9efe2e6f979f1ceb1210c65ed01ec2d4de7f3da63dcec03f96e02ef61ee44bdcb41acc32590
06b9f7bc38778466a2789e219e4f807434a3d19c7c378c74ad9c4d4cd61a93fd6703cda5786759e0b50aedabb22e367cb0e587ab334f938ca99e4
98a7f0be3b350797ba5bba2e90ca3e9cc95f784516632f642b73ceb3ce30eba406c0ad719a3b2bbdb8e600b0a9ee0bb4e9b67853d86ae3b27b6fb
4a3e9fe9b78efb626c943bc72e9bee959507aa9e4e612fab7c89101df520793447cdafaada47ea7753e4d53b5b27c73cc89402bf55e821413ab9f
33159b91502etc7890d8tde6b1668cb417b515c27a21t9cb67455a633tcdt6547b0etc81bt9c6c6ac181de7b23750bebbe3bd24e586c376ct1b1a v
分组 126。129 套户端 分组, 8 服务器 分组, 8 turn(s). 点击选择。
整个对话(2413kB) 🗸 Show data as 原始数据 🗸 🛱
查找: 89504E47 查找下一个(N)
滤掉此流 打印 另存为… 返回CSDN @thatedworking近66

然后复制进去winhex只保留png头尾数据,然后修改文件后缀名为png,就可以得到flag图片

t		
Offset	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ANS	I ASCII
00000000	39 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 %PNG	IHDR
00000016	00 00 06 DA 00 00 09 B0 08 06 00 00 00 14 5F FF Ú °	_Ÿ
00000032	77 00 00 00 06 62 4B 47 44 00 FF 00 FF 00 FF A0 w bKGD	ΫΫΫ
00000048	3D A7 93 00 00 00 09 70 48 59 73 00 00 2E 23 00 3% pH	Ys .#
00000064	DO 2E 23 01 78 A5 3F 76 00 00 00 07 74 49 4D 45 .# x¥?v	tIME
00000080	07 E1 08 15 09 03 29 20 35 7E CB 00 00 00 19 74 á) 5	~Ë t
00000096	45 58 74 43 6F 6D 6D 65 6E 74 00 43 72 65 61 74 EXtCommen	t Creat
00000112	65 64 20 77 69 74 68 20 47 49 4D 50 57 81 0E 17 ed with G	IMPW
00000128	00 00 20 00 49 44 41 54 78 DA EC BD 59 76 ECC 🔂 🕅 @ Harrbauca	ngabe
00000144	ייים ממיל לאה בה היה ביו ביו ביו מים היה היה מים מיי מיי מיי מיי מיי מיי מיי מיי מיי	10 111

tcpxtract

也可以直接

tcpxtract -f 1.pcap

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

0100640200640300640400640500640600640700640300640800640900640A0064060064 0B00640A00640700640800640C00640C00640D00640E00640900640F006716007D0000 6410007D0100781E007C0000445D16007D02007C01007400007C0200830100377D0100 715500577C010047486400005328110000004E696600000696C000006961000000696

选择剪贴板格式	×
UTF-16 Unicode ANSI-ASCII ASCII Hex	_

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	СВ	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
08000000	04	00	64	05	00	64	06	00	64	07	00	64	03	00	64	08	d d d d
00000096	00	64	09	00	64	0A	00	64	06	00	64	0в	00	64	0A	00	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
00000128	0E	00	64	09	00	64	0F	00	67	16	00	7D	00	00	64	10	ddg}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	qUW 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 7B 00 00 69 33 00 ig if i3 00000224 00 00 69 38 00 00 69 35 00 00 69 37 00 00 i8 i5 i7 00000240 00 69 30 00 00 69 32 00 00 69 34 00 00 00 i2 i4 00000256 69 31 00 00 69 65 00 00 69 74 03 00 00 74 i1 ie i i1 ie i t																					
00000224 00 00 69 38 00 00 69 35 00 00 69 37 00 00 i8 i5 i7 00000240 00 69 30 00 00 69 32 00 00 69 34 00 00 00 i2 i4 00000256 69 31 00 00 69 65 00 00 69 7D 00 00 74 i1 ie i} t 00000272 00 00 00 00 74 03 00 00 00 63 68 (t th th 00000288 72 28 03 00 00 74 03 00 00 69 28 00 00 flagt i(t th 00000320 00 00 00 00 73 04 00 00 00 11 i1 igt i1 igt i1 igt i1 i1	00000208	00	00	00	69	67	00	00	00	69	7B	00	00	00	69	33	00	i	.g	i{	i3
00000240 00 69 30 00 00 69 32 00 00 00 69 34 00 00 00 10 12 14 00000256 69 31 00 00 00 69 70 00 00 74 11 1e i) t 00000272 00 00 00 28 01 00 00 74 03 00 00 63 68 (t t ch 00000288 72 28 03 00 00 74 03 00 00 73 74 72 74 04 r(t strt 00000304 00 00 00 00 73 04 00 00 00 01 1 1 1 1 1 p flagt i(s 1.py 00000336 52 03 00 00 00 00 73 04 00 00 00 00 00 1 <t< td=""><td>00000224</td><td>00</td><td>00</td><td>69</td><td>38</td><td>00</td><td>00</td><td>00</td><td>69</td><td>35</td><td>00</td><td>00</td><td>00</td><td>69</td><td>37</td><td>00</td><td>00</td><td>i8</td><td>i</td><td>5</td><td>i7</td></t<>	00000224	00	00	69	38	00	00	00	69	35	00	00	00	69	37	00	00	i8	i	5	i7
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 74 03 00 00 73 74 72 74 04 (t ch 00000288 72 28 03 00 00 00 74 03 00 00 74 03 00 00 69 28 00 00 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 01 00 00 73 04 00 00 00 31 2E 70 79 (s 1.py 00000336 52 03 00 00 00 01 00 01 00 00 73 0A 00 00 00 52 03 H N(R 00000352 48 01 06 01 0D 01 14 01 4E 28 01 00 00 02 28 00 00 H N(R 00000368 00 00 73 04 00 00 31 2E 70 79 74 08 00 00 ((((00000384 00 00 73 04 00 00 31 2E 70 79 74 08 00 00 s 1.pyt 00000400 3C 6D 6F 64 75 6C 65 3E 01 00 00 73 00 00 00 73 00 00 module> s 00000416 00 00 00 31 2E 70 79 74 08 00 00 module> s	00000240	00	69	30	00	00	00	69	32	00	00	00	69	34	00	00	00	i0	i2		i4
00000272 00 00 00 28 01 00 00 74 03 00 00 03 68 (t ch 00000288 72 28 03 00 00 74 03 00 00 73 74 72 74 04 r(t strt 00000304 00 00 00 66 6C 61 67 74 01 00 00 69 28 00 00 flagt i(00000320 00 00 28 00 00 01 00 00 03 12E 70 79 (s lagt 1.py 00000336 52 03 00 00 01 00 00 73 04 00 00 00 01 1.py 00000352 48 01 06 01 00	00000256	69	31	00	00	00	69	65	00	00	00	69	7D	00	00	00	74	i1	ie	i	} t
00000288 72 28 03 00 00 74 03 00 00 07 74 74 74 04 r(t strt 00000304 00 00 00 00 00 00 00 00 66 6C 61 67 74 01 00 00 69 28 00 00 flagt i(00000320 00 00 28 00 00 00 73 04 00 00 00 01 12 70 79 (s 1.py 00000336 52 03 00 00 01 00 00 73 04 00 00 00 00 01 1 1 1 1 28 01 00	00000272	00	00	00	00	28	01	00	00	00	74	03	00	00	00	63	68		(t	ch
00000304 00 00 00 66 6C 61 67 74 01 00 00 69 28 00 00 flagt i(00000320 00 00 28 00 00 00 73 04 00 00 31 2E 70 79 (s 1.py 00000336 52 03 00 00 01 00 00 73 0A 00 00 00 01 0 00 00 00 01 00 00 00 00 00 01 02 00	00000288	72	28	03	00	00	00	74	03	00	00	00	73	74	72	74	04	r (t		strt
00000320 00 00 28 00 00 00 73 04 00 00 31 2E 70 79 (s 1.py 00000336 52 03 00 00 01 00 00 73 0A 00 00 00 01 1 1 1 4E 28 01 00 <td>00000304</td> <td>00</td> <td>00</td> <td>00</td> <td>66</td> <td>6C</td> <td>61</td> <td>67</td> <td>74</td> <td>01</td> <td>00</td> <td>00</td> <td>00</td> <td>69</td> <td>28</td> <td>00</td> <td>00</td> <td>f</td> <td>lagt</td> <td></td> <td>i(</td>	00000304	00	00	00	66	6C	61	67	74	01	00	00	00	69	28	00	00	f	lagt		i(
00000336 52 03 00 00 01 00 00 73 0A 00 00 00 01 R s 00000352 48 01 06 01 0D 01 14 01 4E 28 01 00 00 52 03 H N (R 00000368 00 00 02 28 00 00 00 28 00 00 02 28 00 00 28 00 00 28 00 00 00 00 (00000320	00	00	28	00	00	00	00	73	04	00	00	00	31	2E	70	79	(s		1.py
00000352 48 01 06 01 00 01 14 01 4E 28 01 00 00 52 03 H N(R 00000368 00 00 00 28 00 00 00 28 00 00 02 28 00 00 28 00 00 28 00 00 28 00 00 28 00 00 28 00 00 00 28 00 00 00 28 00 00 00 28 00 00 00 28 00 00 00 28 00 <td>00000336</td> <td>52</td> <td>03</td> <td>00</td> <td>00</td> <td>00</td> <td>01</td> <td>00</td> <td>00</td> <td>00</td> <td>73</td> <td>0A</td> <td>00</td> <td>00</td> <td>00</td> <td>00</td> <td>01</td> <td>R</td> <td></td> <td>s</td> <td></td>	00000336	52	03	00	00	00	01	00	00	00	73	0A	00	00	00	00	01	R		s	
00000368 00 00 00 28 00 00 02 8 00 00 28 00 00 02 8 00 00 28 00 00 02 8 00 00 28 00 00 02 28 00 00 02 28 00 00 02 28 00 00 02 28 00 00 02 28 00 00 02 28 00 00 02 28 00 00 02 28 00 00 02 28 00	00000352	48	01	06	01	0D	01	14	01	4E	28	01	00	00	00	52	03	Н		N (R
00000384 00 00 73 04 00 00 31 2E 70 74 08 00 00 00 s 1.pyt 00000400 3C 6D 6F 64 75 6C 65 3E 01 00 00 73 00 00 00 cmodule> s 00000416 00	00000368	00	00	00	28	00	00	00	00	28	00	00	00	00	28	00	00	(((
00000400 3C 6D 6F 64 75 6C 65 3E 01 00 00 073 00 00 00 <module> s CSDN @Hardworking666</module>	00000384	00	00	73	04	00	00	00	31	2E	70	79	74	08	00	00	00	s	1	·py	t
CSDN @Hardworking666	00000400	3C	6D	6F	64	75	6C	65	3E	01	00	00	00	73	00	00	00	<mod< td=""><td>lul,e></td><td></td><td>S</td></mod<>	lul,e>		S
0000410 00	00000416	00														C	.5DI\	I @Ha	rawa	orkir	igobo

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

ryakalifisher:~/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'. `\Document usage: stegosaurus.py [- ca	ts\CTF\py\stegosaurus-master>python stegosaurus.py -h -h] [-p PAYLOAD] [-r] [-s] [-v] [-x] [-e EXPLODE] arrier
positional arguments: carrier	Carrier py, pyc or pyo file
optional arguments: -h,help -p PAYLOAD,payload	show this help message and exit PAYLOAD Embed payload in carrier file
-s,side-by-side	Do not overwrite carrier file, install side by side instead.
-v,verbose	Increase verbosity once per use
-x,extract -e EXPLODE,explode	EXTRACT payload from carrier file EXPLODE Explode payload into groups of a limited length if Decessary 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{yOud1any1s1}

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 127=01111111 191=10111111 255=1111111 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:
       a = '00'
       a = '10'
 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,与维吉尼亚密码密码类似,但使用不同的方法生成密钥。