# 攻防世界——MISC 高手区题解



 11 篇文章 7 订 订阅专栏
 目录

- 1, base64stego
- 2, easycap
- 3, Avatar
- 4, What-is-this
- 5, 签到题
- 6, Training-Stegano-1
- 7, Excaliflag
- 8, Get-the-key.txt
- 9, glance-50
- 10, 4-2
- 11,misc1
- 12, embarrass
- 13, 肥宅快乐题
- 14, 小小的PDF

至此 我在攻防世界中已经 遇到 三种 pdf 的隐写方式:

- 15, Cephalopod
- 16, hit-the-core
- 17, pure\_color
- 18, 2-1
- 19, János-the-Ripper
- 20, 2017\_Dating\_in\_Singapore
- 21, 4-1

22, 神奇的Modbus

- 23, 5-1
- 24, can\_has\_stdio?
- 25, MISCall

26, 3-1

- 27,适合作为桌面
- 28, banmabanma
- 29,我们的秘密是绿色的
- 30, simple\_transfer
- 31, Just-No-One
- 32, warmup
- 33, Erik-Baleog-and-Olaf
- 34, Py-Py-Py
- 35, reverse\_it
- 36, mysql

### 1, base64stego

### 很神奇的base 64 加密 (在base 64 的密文中加密,还不影响原明文)

(转载于: https://www.tr0y.wang/2017/06/14/Base64steg/ 做了一定的修改)

(简要原理, ascii码是用8位二进制表示一个字符的, 而base64是用6位二进制表示一个字符, 将明文字 符转化为二进制后再每6位划分成一个"字节", 然后将每个字节转化为一个字符, 就变成了base64密文, 而在base64的密文中加密是利用, 每一段密文的最后4位二进制是不影响明文的, 可以将flag转化为二进 制后拆分隐藏在每一段的最后4位二进制中)

复习一下 Base64 吧

BASE64 是一种编码方式, 是一种可逆的编码方式. 编码后的数据是一个字符串, 包含的字符为: A-Za-z0-9+/ 共 64 个字符: 26 + 26 + 10 + 1 + 1 = 64 其实是 65 个字符, "="是填充字符.

64个字符需要6位二进制来表示,表示成数值为0~63.

Value	Char	Value	Char	Value	Char		Value	Char
0	A	16	Q	32	g		48	W
1	В	17	R	33	h		49	x
2	С	18	S	34	i		50	у
3	D	19	Т	35	j		51	z
4	E	20	U	36	k		52	0
5	F	21	V	37	1		53	1
6	G	22	¥	38	m		54	2
7	H	23	X	39	n		55	3
8	I	24	Y	40	0		56	4
9	J	25	Z	41	р		57	5
10	K	26	a	42	q		58	6
11	L	27	Ъ	43	r		59	7
12	M	28	с	44	s		60	8
13	N	29	d	45	t		61	9
14	0	30	е	46	u		62	+
15	Р	31	图片	来ള:	WWW	V.	tr <b>6</b> 8/.v	vang

这样,长度为3个字节的数据经过 Base64 编码后就变为4个字节

### 编码

比如,字符串"Tr0"经过 Base64 编码后变为"VHW"

Т								r						0									
0	1	0	1	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0	0	0
		١	<b>V</b> Н						图片来自:www.tr₩y.wang								ng						

上面说的字符串长度为 3 个字节的数据位数是 8x3=24, 可以精确地分成 6x4. 如果字节数不是 3 的倍数, 则位数就不是 6 的倍数, 那么就不能精确地划分成 6 位的块. 此时, 需在原数据二进制值后面添加零, 使其字节数是 6 的倍数. 然后, 在编码后的字符串后面添加 1 个或 2 个等号"=", 表示所添加的零值字节数. 比如, 字符串"TrOy"经过 Base64 编码后变为"VHlweQ=="



橙色底纹就是添加的 0.

这是 Base64 编码的方式.

#### 解码

解码就是编码的逆过程.

- 1. 把 Base64 字符串去掉等号,转为二进制数(VHlweQ== -> VHlweQ -> 0101010001110010001110000011110010000).
- 2. 从左到右, 8 个位一组, 多余位的扔掉, 转为对应的 ASCII 码(01010100 01110010 00110000 01111001 0000 -> 扔掉最后 4 位 -> 01010100 01110010 00110000 01111001 -> TrOy)

base64密文中隐写原理

注意红色的 0, 我们在解码的时候将其丢弃了, 所以这里的值不会影响解码. 所以我们可以在这进行隐写.

为什么等号的那部分0不能用于隐写?因为修改那里的二进制值会导致等号数量变化,解码的第1步会受影响. 自然也就破坏了源字符串.

而红色部分的 0 是作为最后一个字符二进制的组成部分, 还原时只用到了最后一个字符二进制的前部分, 后面的 部分就不会影响还原.

唯一的影响就是最后一个字符会变化.如下图



如果你直接解密'VHlweQ=='与'VHlweR==',得到的结果都是'TrOy'.

当然, 一行 base64 顶多能有 2 个等号, 也就是有 2\*2 位的可隐写位. 所以我们得弄很多行, 才能隐藏一个字符串, 这也是为什么题目给了一大段 base64 的原因.

接下来, 把要隐藏的 flag 转为 8 位二进制, 塞进去就行了.

加密:

```
转载自: https://www.tr0y.wang/2017/06/14/Base64steg/
import base64
flag = 'Tr0y{Base64isF4n}' #flag
bin_str = ''.join([bin(ord(c)).replace('0b', '').zfill(8) for c in flag])
base64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('0.txt', 'rb') as f0, open('1.txt', 'wb') as f1: #'0.txt'是明文, '1.txt'用于存放隐写后的 base64
for line in f0.readlines():
    rowstr = base64.b64encode(line.replace('\n', ''))
    equalnum = rowstr.count('=')
    if equalnum and len(bin_str):
        offset = int('0b'+bin_str[:equalnum * 2], 2)
        char = rowstr[len(rowstr) - equalnum - 1]
        rowstr = bin_str[equalnum*2:]
    f1.write(rowstr + '\n')
```

解密:

```
import base64
b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('stego.txt', 'rb') as f: #stego.txt 为在base64密文中加密后的密文
flag = ''
bin_str = ''
for line in f.readlines():
    stegb64 = str(line, "utf-8").strip("\n")
    rowb64 = str(base64.b64encode(base64.b64decode(stegb64)), "utf-8").strip("\n")
    offset = abs(b64chars.index(stegb64.replace('=', '')[-1]) - b64chars.index(rowb64.replace('=', '')[
    equalnum = stegb64.count('=') # no equalnum no offset
    if equalnum:
        bin_str += bin(offset)[2:].zfill(equalnum * 2)
for i in range(0, len(bin_str), 8):
    print(chr(int(bin_str[i:i + 8], 2)),end='')
```

给出加密脚本执行结果



flag{Base\_sixty\_four\_point\_five}

### 2, easycap

下载下来是一个截取的流量包,用wireshark打开,先追踪流,没想到直接得到了flag:



🗲 Wireshark · 追踪 TCP 流 (tcp.stream eq 0) · 32b6f366cae544cd85008e5fdd56155a.pcap

FLAG: 385b87afc8671dee07550290d16a8071

### 3, Avatar

是一张图片,先用formost分离一下,没有什么东西,用stegsolve看一下也没发现什么,再放进winhex中看一下,依旧没收获,

再试其他工具,.....在用outguess时分离出来了隐藏信息:



We should blow up the bridge at midnight 直接提交

### 4, What-is-this

解压文件,是一个没有后缀的文件,放进winhex中审查一下,发现有几个文件名,目测这是一个压缩包,把后 缀给为zip后解压,



两张图片,正常思路: 1,图片拼接 2,盲水印 3,各有一部分flag

先试一下 图片拼接,用stegsolve把两张图片合成一下:

*		_		×
XOR				2 13 1 2 3 1 4 Mar 1 2 2 2
AZA	DI	TC	)WI	ER
	¶ps://b	nog.cs	\$aveat/vh	kjhwbs

### 5, 签到题

(本以为很简单,实际.....)

将Z2dRQGdRMWZxaDBvaHRqcHRfc3d7Z2ZoZ3MjfQ== base64解密一下: ggQ@gQ1fqh0ohtjpt\_sw{gfhgs#}

你以为这就结束了? 提交错误

猜测{}的位置不对,进行一下栅栏密码:ggQ@gQ1fqh0ohtjpt\_sw{gfhgs#}

2字一栏: gQg1q0hjts{fg#g@Qfhotp\_wghs} 4字一栏: ggqht{ggQht\_gsQ10jsf#@fopwh} 7字一栏: gfjggqpfQhth@0\_ggossQhw#1t{} 14字一栏: gjgpQt@\_gsQw1{fgqfhh0gosh#t} 就4个都提交一下试试,结果全都不对,,,,

用 ggqht{ggQht\_gsQ10jsf#@fopwh} 再试一下 凯撒密码:

4字一栏: ggqht{ggQht\_gsQ10jsf#@fopwh} 解密 使用英文字典智能分 第1次解密:4字一栏: ggqht{ggqht\_gsq10jsf#@fopwh} 第2次解密:4字一栏:ffpgs{ffpgs\_frp10ire#@enovg} 第3次解密:4字一栏: eeofr{eeofr\_eqo10hqd#@dmnuf} 第4次解密:4字一栏: ddneq{ddneq\_dpn10gpc#@clmte} 第5次解密:4字一栏: ccmdp{ccmdp\_com10fob#@bklsd} 第6次解密:4字一栏: bblco{bblco\_bnl10ena#@ajkrc} 第7次解密:4字一栏: aakbn{aakbn\_amk10dmz#@zijqb} 第8次解密:4字一栏: zzjam{zzjam\_zlj10cly#@yhipa} 第9次解密:4字一栏: yyizl{yyizl\_yki10bkx#@xghoz} 第10次解密:4字一栏: xxhyk{xxhyk\_xjh10ajw#@wfgny} 第11次解密:4字一栏: wwgxj{wwgxj\_wig10ziv#@vefmx} 第12次解密:4字一栏: vvfwi{vvfwi\_vhf10yhu#@udelw} 第13次解密:4字一栏: uuevh{uuevh\_uge10xgt#@tcdkv} 第14次解密:4字一栏: ttdug{ttdug\_tfd10wfs#@sbcju} 第15次解密:4字一栏: ssctf {ssctf\_sec10ver#@rabit} 第16次解密:4字一栏: rrbse{rrbse\_rdb10udq#@qzains] 第17次解密:4字一栏: qqard{qqard\_qcal0tcp#@pyzgr} 第18次解密:4字一栏: ppzqc{ppzqc\_pbz10sbo#@oxyfq} 第19次解密:4字一栏: ooypb{ooypb\_oay10ran#@nwzep} 第20次解密:4字一栏: nnxoa{nnxoa\_nzx10qzm#@mvwdo}

看到一个ctf直接提交:

ssctf{ssctf\_sec10ver#@rabit}

### 6, Training-Stegano-1

好小的一张图片,第一反应是用winhex修改宽和高,没想到打开直接发现了flag: 0 42 4D 66 00 00 00 00 00 00 00 36 00 00 00 28 0.0 BMf 6 ( 00 00 04 00 00 00 04 00 00 00 01 00 18 00 00 00 0 00 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0 6B 20 77 61 74 :0 00 00 00 00 00 00 4C 6F 6F 68 20 Look what 74 68 65 20 68 65 78 64 69 74 20 72 65 76 0 2D 65 the hex-edit rev 0 65 61 6C 65 64 3A 20 70 61 73 73 77 64 3A 73 74 ealed: passwd:st 65 67 61 6E 6F 49 10 eganoI

steganol

# 7, Excaliflag

用foremost分离没发现东西,用stegsolve看一下:



# 8, Get-the-key.txt

解压文件 得到一个 没有后缀的 文件 forensic100,看这个名字 我还以为是一个 截取的流量包,后缀改为 pcap 看了一下,没发现什么的,用winhex看一下,猜测是一个压缩包,后缀改为 zip

解压得到 key.txt:

SECCON{@]NL7n+-s75FrET]vU=7Z}

### 9, glance-50

打开是一个极窄的git图片,那还还用想吗?思路肯定是,先把动态图的每一帧分离出来,再拼接起来

上脚本:

分离:

```
import os
from PIL import Image
def seoaration_gif(gif_file):
   png_dir = gif_file[:-4] + '/'
   os.mkdir(png_dir)
   img = Image.open(gif_file)
   try:
        while True:
            current = img.tell()
            img.save(png_dir+str(current)+'.png')
            img.seek(current+1)
   except:
        pass
if __name__=='__main__':
   gif_file = '123.gif'
   seoaration_gif(gif_file)
```

### 拼接:

```
#拼接图像.py
from PIL import Image
path = "C:\\Users\\HP\\Desktop\\文档\\Python\\text\\123\\"
save_path = 'C:\\Users\\HP\\Desktop\\文档\\Python\\text\\'

im = Image.new('RGBA',(2*201,600)) #创建新照片
imagefile = [] #存储所有的图像的名称
width = 0
for i in range(0,201):
    imagefile.append(Image.open(path+str(i)+'.png')) #遍历,将图像名称存入imagfile
for image in imagefile:
    im.paste(image,(width,0,2+width,600)) #将图片张贴到另一张图片上
    width = width +2
im.save(save_path+'result.png')
im.show()
```

得到图片:



### 10, 4-2

得到一段看起来毫无规律的密文:

Eg qnlyjtcnzydl z umaujejmjetg qeydsn eu z bsjdtx tw sgqtxegc al kdeqd mgeju tw yrzegjsoj zns nsyrzqsx kejd Ew ltm fgtk jds kzl tw sgqtxegc m kerr csj jds wrzc kdeqd eu qrzuueqzr-qeydsn\_eu\_gtj\_usqmnejl\_du

不知道从何下手,看了一下别人的writeup,发现是词频分析

### https://quipqiup.com/



flag{classical-cipher\_is\_not\_security\_hs}

# 11,misc1

打开是一段密文:

以为是base64,发现解不出来,再回头仔细看,发现只有 0~9 a~f 十六进制

但是,直接用16进制=》ascii发现是一堆乱码,

常识 ascii 只有 到128 , 而发现这一串16进制每两位化成 十进制 后都大于 128

肯定得减去128后再转成 ascii:

直接上代码:

```
char = "d4e8e1f4a0f7e1f3a0e6e1f3f4a1a0d4e8e5a0e6ece1e7a0e9f3baa0c4c4c3d4c6fbb9e1e6b3e3b9e4b3b7b7e2b6b1e4b2b
for i in range(0,len(char),2):
    print(chr(int(char[i:i+2],16)-128),end="")
```

C:\Users\HP\AppData\Local\Programs\Python\Python37\python3. exe C:/Users/ That was fast! The flag is: DDCTF{9af3c9d377b61d269b11337f330c935f} Process finished with exit code 0

That was fast! The flag is: DDCTF{9af3c9d377b61d269b11337f330c935f}

#### 12, embarrass

下载下来是一个 流量包, 在linux中 搜索一下 flag 字符串 看会不会有意外收获:

```
luohao@ubuntu:~/Desktop$ strings './misc_02.pcapng' | grep flag
GET /flag.php HTTP/1.1
GET /
        .doc HTTP/1.1
   Good_b0y_W3ll_Done}
   {Good_b0y_W3ll_Done}
{Good_b0y_W3ll_Done}
   {Good b0y W3ll Done}
     Some antivirus programs mistake XAMPP for a virus, typically flagging the file xampp-manage
\cdot.exe This is a false positive meaning that the antivirus erroneously identified it as a virus, when
it is not. Before we release each new version of XAMPP we run it through virus scanning software. A
t the moment we are using <a href="http://www.kaspersky.com/virusscanner">Kapersky Online Virus Scan
ner</a>. You can also use the online tool <a href="https://www.virustotal.com/">Virus Total</a> for
scanning XAMPP or send us an email to security (at) apachefriends (dot) org if you find any issue.</
D>
filter.default_flags<i>no value</i><i>no val
ue</i>
 tr>windows_tracing_flags 3 
                                                                      https://blog.csdn.net/vhkjhwbs
   hao@ubuntu:~/DesktonS
```

没想到直接出来了

flag{Good\_b0y\_W3ll\_Done}

### 13, 肥宅快乐题

a.这个题目下载到手是一个.swf后缀的文件 b.用到一个特殊的工具,叫做potplayer c.把这个.swf文件使用这个工具打开 d.根据提示注意对话就好 点击 右下角的 开始 就会 自动播放 在 放到 57 帧 的时候, 出现了 下面的对话 有一段 base64:



狗日的 不能复

提取出来: U1IDe0YzaVpoYWIfa3U0aWxIX1QxMTF9

base64解码得flag: SYC{F3iZhai\_ku4ile\_T111}

### 14,小小的PDF

下载下来是一个 pdf文件,不管有没有隐藏文件,先用formost 分离一下,没想到直接 分离出来了三张图片,包含一张flag图片:



https://blog.csdn.net/vhkjhwbs

# SYC{so\_so\_easy}

至此 我在攻防世界中已经 遇到 三种 pdf 的隐写方式:

1,新手区: pdf:将flag隐藏在图片的下面 》 需要 先将格式转为 word 再将图片拖开 就能看到 flag

2, 新手区: stegano:利用类似水印的方法将flag隐藏在大量文字下面(不清粗具体方法)》 全选复制 到 txt文件中就能 显示出密文

# 15, Cephalopod

打开数据包,尝试性 在分组字节流中 搜索字符串 flag 然后就看到了有一张 flag.png:

🗲 Cephalopod.pcap				- 0	×
文件(E) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S)	电话(Y) 无线(W) 工具(I) 帮助()	1)			
◢ ■ ∅ ◎ ▮ 🖹 🗙 🖏 ♀ ⇔ 🛎 ∓ 🖢 📃 🚍	⊕, ⊖, ⊝, ™				
■ 应用显示过滤器 … 〈Ctrl=/〉	Jana				対 +
分组字节流 ∨ 宽窄 ∨ □区分大小写	字符串 ~ flagS			查找回	取消
No. Time Source	Destination	Prot	ocol Le	ngth Info	^
70 15:18:43.372323 10.0.2.7	10.0.2.10	Cepł	1	169 Client Session	
71 15:18:43.373461 10.0.2.10	10.0.2.7	Cepł	n	178 ACK   Client Session	
72 15:18:43.373492 10.0.2.7	10.0.2.10	TCP		66 54924 → 6812 [ACK] Seq=1367 Ack=1704 Win=32768 Len=0 TSval=2	3
73 15:18:43.580082 10.0.2.7	10.0.2.10	Cepł	n	75 ACK	
74 15:18:43.618204 10.0.2.10	10.0.2.7	TCP		66 6812 → 54924 [ACK] Seq=1704 Ack=1376 Win=35456 Len=0 TSval=2	7 🕑
75 15:18:44.869235 10.0.2.7	10.0.2.10	Cepł	า	315 Client Request	
76 45 40 44 060704 40 0 2 40	40.0.2.7	TCD		co coto - Front Facel care and and acor the octoo tar o round o	
Tag: message (0x07)		^	0000	08 00 27 4a 44 a0 08 00 27 da 21 48 08 00 45 00 ··· 'JD··· '·!	H··E·
> Message Header, Type: C_CEPH_MSG_CLIENT_R	EQUEST, From: client4137	, Fro	0010	01 2d 25 43 40 00 40 06 fc 77 0a 00 02 07 0a 00 -%C@ @ w	
✓ Client Request, Operation: MDS_OP_CREATE			0020	02 0a d6 8c 1a 9c 62 45 6e 70 52 4b 9e bb 80 18 ·····bE npR	K
Oldest TID: 3			0030	01 00 19 30 00 00 01 01 08 0a 00 00 5d 16 00 040	.]
MDS Map Epoch: 5			0040	22 92 07 06 00 00 00 00 00 00 03 00 00 00 00 00	
Flags: 0x02			0050	00 00 00 18 00 /f 00 02 00 ae 00 00 00 00 00 00	
Number of Retries: 0			0000	00 00 00 00 00 00 00 08 29 10 00 00 00 00 00 00 00 00 00 00 00 00	
Number of Forwards: 0			0070		
Number of Releases: 1			0000		
Operation: MDS OP CREATE (0x00001301)			0020		
Caller User ID: 0			00b0	00 00 00 00 00 00 00 00 00 00 00 00 00	
Caller Group ID: 0			00c0	00 00 00 00 00 00 00 00 00 00 00 00 00	
Inode: 0			00d0	01 01 00 00 00 00 00 00 00 08 00 00 00 <u>66 6c 61</u>	••fla
✓ Path. Inode: 0x0000000000000001. Rel: '	flag.png"		00e0	67 2e 70 6e 67 01 00 00 00 00 00 00 00 00 00 00 00 00	····· )
Encoding Version: 0x01			00f0	00 00 01 00 00 00 00 00 00 03 00 00 00 00 00 🔪	
Trode: 0x00000000000000			0100	00 00 55 00 00 00 55 03 00 00 02 00 00 02 00 💴 U ····	
<pre>     Relative component: flag.nng </pre>			0110	00 00 00 00 00 00 00 00 00 00 00 00 00	···т,
Size: 8			0120	9d 59 55 55 87 33 3c 77 9f 0c 00 00 00 00 00 ·YUU·3 <w< td=""><td>•••••</td></w<>	•••••
Data: flag.png			0130	00 00 6d 23 b1 6b 9e a4 9e 7c 05	
x Second Path		~			
<		>			
● ℤ Data (ceph.string.data), 8 字节				分组: 339 · 已显示: 339(100.0%) <sup>https://blog/</sup> P截置:	)efault

然后 追踪流 (还原原始数据包)发现flag.png 的数据并不在 75 这个包里:



看下一个流:发现了 png 的数据文件:选择 原始数据流 =》save as flag.png (直接保存的话,部分数据会丢失)

Wireshark · 追踪 TCP 流 (tcp.stream eq 2) · Cephalopod.pca	p			- 🗆
	+	1		
.0000000001.00000000GG.				
*	N	B.QeA		
······B·····B·····	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	*		
5T.Z		.10000000001.0000000	)G	
5				
		*		'
*	#¢)	c (p		
ΥΥ	····/···/		)"	
HDR	pHYs##.x.?v.	tIME) 5~	tEXtComment.Created	with GIMPW
::xF.O.U:.w.=BT.bi>aG			31.11.	
s.svT7Tu? .~>q.OjSt.SygGi.<.	.*lx}k.].y	i*s0e	e.r]oUt0".?y.I	NO+
46nV.^V8UW1	}"			
.'Dy!.eMcna.DA%	8w.f.x.!.0.t4	7.tMLgxgY		
6 30IPy{Q†2.B.<	<@1			
	v466.w5	II. [A:1V		
.k.h.A{Q\'tU.3eG7P\$	.vCcl+nlQ8gE.[	).		
&~aee4HtQB].{7	≻G].)_8*.Tu.V	/.t:=V3.)jf	[`xMt;	
.1zB(.KM.:'8t1).x*{	2yb			
/. R				
1W				
9 <i>第 戶間 分組, 3 服务服 分組, 3 turn(s).</i>				
各个对话(2413 kB)	示和保存数据为 ASCII V			潼
44-	ASCII			
(14):	EBCDIC			<u>■</u> 扱い一小
	Hex 特術 UTF-8	滤掉此流 打印	Save as… 返回	Close Help
	UTF-16			
7	原始数据	5		

### 然后用 winhex 打开, 删除 图片头前面的 无关数据:

00	00	14	00	00	00	24	00	00	00	05	A2	9A	59	00	00	Ş ¢šY
00	00	00	00	00	00	00	00	00	00	00	00	00	00	04	04	
10	00	00	00	01	00	00	00	00	00	00	00	FF	FF	FF	FF	ŸŸŸŸ
00	00	00	00	01	01	00	00	00	00	00	00	00	47	07	9E	GŽ
0C	$\mathbf{F}\mathbf{F}$	FF	FF	FF	14	00	00	00	31	30	30	30	30	30	30	<u>ÿÿÿÿ</u> 1000000
30	30	30	31	2E	30	30	30	30	30	30	30	30	01	00	01	0001.00000000
22	00	00	00	00	00	00	00	00	00	00	00	00	23	CB	24	"#Ê\$
00	00	00	00	00	FF	FF	$\mathbf{F}\mathbf{F}$	FF	FF	FF	FF	FF	01	00	00	<u> </u>
00	23	CB	24	00	FE	FF	$\mathbf{F}\mathbf{F}$	FF	FF	FF	FF	FF	01	00	00	#Ë\$ þ <u>ÿÿÿÿÿÿ</u> ÿ
00	00	00	00	00	00	00	00	00	01	00	00	00	89	50	4E	%PN
47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	00	00	06	G IHDR
DA	00	00	09	в0	08	06	00	00	00	14	5F	$\mathbf{F}\mathbf{F}$	77	00	00	ΰ°_ÿw
00	06	62	4B	47	44	00	$\mathbf{F}\mathbf{F}$	00	$\mathbf{F}\mathbf{F}$	00	FF	<b>A</b> 0	BD	Α7	93	bKGD ÿ ÿ ÿ ½§"
00	00	00	09	70	48	59	73	00	00	2E	23	00	00	2E	23	pHYs .# .#
01	78	A5	ЗF	76	00	00	00	07	74	49	4D	45	07	E1	80	x¥?v tIME á
15	09	03	29	20	35	7E	СВ	00	00	00	19	74	45	58	74	) 5~Ë tEXt
43	6F	бD	6D	65	6E	74	00	43	72	65	61	74	65	64	20	Comment Created
77	69	74	68	20	47	49	4D	50	57	81	0E	17	00	00	20	with GIMPW
00	49	44	41	54	78	DA	EC	BD	59	76	EC	CA	8E	24	EA	IDATxÚì½YvìÊŽ\$ê
D0	00	в2	DE	9в	FF	60	51	1F	5в	21	91	0C	92	6E	66	а⊉>ÿ`Q [!`'nf
					-		-		_		-			-		

然后就看到了flag.png:



HITB{95700d8aefdc1648b90a92f3a8460a2c}

# 16, hit-the-core

下载下来是一个 core文件:

小知识点 core:

**core** 文件: 核心文件(core file),也称核心转储(core dump),是操作系统在进程收到某些信号而终止运行时,将此时进程地址空间的内容以及有关进程状态的其他信息写出的一个磁盘文件。这种信息往往用于调试。

核心文件通常在系统收到特定的信号时由操作系统生成。信号可以由程序执行过程中的异常触发,也可以由外 部程序发送。动作的结果一般是生成一个某个进程的内存转储的文件,文件包含了此进程当前的运行堆栈信 息。有时程序并未经过彻底测试,这使得它在执行的时候一不小心就会遭到破坏。这可能会导致核心转储 (core dump)。现在的UNIX系统极少会面临这样的问题。即使遇到,程序员可以通过核心映像调试程序来找 到错误原因。

不知如何下手,看别人的博客,发现是用 strings 命令 提取出 文件中的 字符串:

strings 123.core

[]A\A]A^A\_ cvqAeqacLtqazEigwiXobxrCrtuiTzahfFreqc{bnjrKwgk83kgd43j85ePgb\_e\_rwqr7fvbmHjklo3tews\_hmkogooyf0vbnk0ii87D rfgh\_n kiwutfb0ghk9ro987k5tfb\_hjiouo087ptfcv} ;\*3\$" (q9e aliases

竟然能从这一段字符串中发现几个大写字母之间的联系: ALECTF 而且每个字母中间隔了 5个字符,

提取这些字符: (真是TM服了,这题都能出)

```
data = 'cvqAeqacLtqazEigwiXobxrCrtuiTzahfFreqc{bnjrKwgk83kgd43j85ePgb_e_rwqr7fvbmHjklo3tews_hmkogooyf0vbnk0
flag = ''
for i in range(3, len(data), 5):
    flag += data[i]
print(flag)
```

得到flag: ALEXCTF{K33P\_7H3\_g00D\_w0rk\_up}

### 17, pure\_color

打开是一张白色图片,放进stegsolve中分析一下:

# Flag is true\_steganographers\_doesnt\_need\_any\_tools

< >

flag{true\_steganographers\_doesnt\_need\_any\_tools}

# 18, 2-1

是一个png图片,打不开,文件损坏,用 winhex 打开发现 文件头 是错的,把文件头修改为 89 50 4E 47 保存后发现还是打不开,用 tweakpng 打开 有报错:

确定

Incorrect crc for IHDR chunk (is 932f8a6b, should be 55d5f64f)

把文件 IHDR 的crc校验值 修改为 55 d5 f6 4f 后发现 还是打不开,

再仔细看发现图片的宽是0

根据 crc校验值 算宽是多少:

```
import struct
import binascii
import os

m = open("misc4.png","rb").read()
for i in range(1024):
    c = m[12:16] + struct.pack('>i', i) + m[20:29]
    crc = binascii.crc32(c) & 0xffffffff
    if crc == 0x932f8a6b: #自己根据情况改
        print(i)
```

得到宽度为709,修改宽度为709换算为16进制为 02 c5

C2c\_u\_kn0W}

得flag

wdflag{Png\_C2c\_u\_kn0W}

# 19, János-the-Ripper

解压后得到一个 没有后缀的文件 放进winhex中发现是一个 zip包,后缀改为 zip

解压发现需要密码,直接爆破得到密码: fish

0			10			
ARCHPR 4.54	4 - 50%			1000		×
文件(F) 恢复(R)	) 帮助(H)					
8° . R	8 🔒	Ì	-	2		
打开 开如	自 停止	基准则试	升级	帮助	关于	退出
加密的 ZIP/RAR	/ACE/ARJ 文件		攻击类型			
C:\Users\HP\Des	ktop\misc100.z	ip	暴力			$\sim$
	口公已成功恢复	<b>3</b> 1				
范围长度						
暴力范围选项		Advance	d Archive Pas	sword Reco	wery 统计(	言息:
✓ 所有大写技	总计口令			7,763,827		
✓ 所有小与拉	总计时间			326ms		
所有特殊符	平均速度(口:	令/秒)		23,815,420	0	
空格	这个文件的口	1会		fish		
	十六讲制口会	>		66 69 73 6	в	
状态窗口						
2019/9/15 17:0		- 保存			<b>V</b>	确定
2019/9/15 17:0	46 - D&Dd	市版官口				
2019/9/15 17:03	:46 - 'fish' 是这	个文件的一	个有效口令			~
当前口今:	fish		亚均速度:	24.3	261.959 n/s	
已用时间:			剩余时间:	2 17.		
口令长度 = 4, 总i	+: 14,776,336	已处理: 7,5	21,593			
		50	%			
ARCHPR version 4.	54 (c) 1997-201	2 ElcomSoft (	Co. Ltd.			

解压得到flag:

flag{ev3n::y0u::bru7us?!}

# 20, 2017\_Dating\_in\_Singapore

脑洞是真尼玛大!

12段 数字,每一段都是有偶数个数字,发现如果两个字符一组的话,最大的数字的31,

刚好是日历中最大的数字,然后就在日历上一个数字一个数字地连,然后就画出了下面的字符:



flag:

HITB{CTFFUN}

### 21, 4-1

解压得到一张图片,用formost分离出来一个 压缩包,解压得到

得到两张"一样"的图片,和一个 tips.txt 大意是指 day2.png 中比 day1.png 多了点"东西"

这一看肯定是 水印啊, 盲水印啊,

直接提取盲水印:

python2 bwm.py decode day1.png day2.png flag.png

(c) 2017 Microsoft Corporation。保留所有权利。 D:\谷歌下载\e7dede5e409641239b36e0d7d360368c\outfile\zip\00000811\day2's secret>python2 bwm.py decode day1.png day2.png flag.png image<day1.png> + image(encoded)<day2.png> -> watermark<flag.png> D:\谷歌下载\e7dede5e409641239b36e0d7d360368c\outfile\zip\00000811\day2's secret>\_

https://blog.csdn.net/vhkjhwb

得到水印:



### 22, 神奇的Modbus

Modbus是一种串行通信协议,

Modbus协议当前存在用于串口、以太网以及其他支持互联网协议的网络的版本。

大多数Modbus设备通信通过串口EIA-485物理层进行

用wireshark 打开流量包,直接在 分组字节流 中 搜索 sctf 字符串 直接出来:

			_,,	
192.168.130.1	Modbus	103 Response: Trans:	1; Unit: 1, Fu	nc: 3: Read Holding Re…
192.168.130.130	TCP	54 62234 → 502 [ACK]	Sea=23353 Ack=32694	Win=525312 Len=0
	0000	00 50 56 c0 00 08 00	0c 29 02 23 7c 08	00 45 00 ·PV···· )·# ··E·
	0010	00 59 7c 1f 40 00 40	06 38 ab c0 a8 82	82 c0 a8 ·Y @ @ 8 · · · · ·
ng Registers (3)	0020	82 01 01 f6 f3 1a ab	23 64 c6 fd ff 9f	46 50 18 ·····# d····FP·
	0030	00 e5 c9 6a 00 00 00	01 00 00 00 2b 01	03 28 00 ··· j···· +·· (·
15]	0040	73 00 63 00 74 00 66	00 7b 00 45 00 61	00 73 00 s c t f {Eas
x2]	0050	79 <u>00 5f</u> 00 4d 00 64	00 62 00 75 00 73	00 7d 00 ( yM.d. b.u.s.}.   `
	0060	00 00 00 00 00 00 00 00	)	· · · · · · · · · · · · · · · · · · ·
				https://blog.csdn.net/vhkjhwt.s

但是 sctf{Easy\_Mdbus} 竟然不是 正确答案

正确答案: sctf{Easy\_Modbus}

# 23, 5-1

是一个不知道文件类型的 文件 , 用 file 分析一下 显示是类型 为 data ,

看别人的博客,大意是需要用 xortool 工具

先尝试出 key: GoodLuckToYou



其中-l就是指定密钥长度,-c表示出现频率最高的字符。这个需要根据经验,比如文本内容一般是空格 (20),二进制文件一般是00

然后用 脚本进行 解密:

```
import os
```

```
c = open("123",'rb').read()
key = "GoodLuckToYou"
def xor(c,k):
    keylen = len(k)
    res = ""
    for pos,c in enumerate(c):
        res +=chr(ord(c) ^ ord(k[pos % keylen]))
    return res
print xor(c,key)
```

E: C:\Windows\System32\cmd.exe	—		×
Microsoft Windows [版本 10.0.16299.1004] (c) 2017 Microsoft Corporation。保留所有权利。			^
D:\谷歌下载>python2 w.py The opening line of the novel famously announces: "It is a truth universally acknowledged, that a single man i f a good fortune must be in want of a wife." This sets marriage as a central subject欽構nd really, a central p he novel generally. Readers are poised to question whether or not these single men are, in fact, in want of a ch desires are dictated by the "neighbourhood" families and their daughters who require a "good fortune". Marr lex social activity that takes political economy, and economy more generally, into account. In the case of Cha for example, the seeming success of her marriage lies in the comfortable economy of their household, while the between Mr and Mrs Bennet serves to illustrate bad marriages based on an initial attraction and surface over s omic and psychological). The Bennets' marriage is one <u>such example that the youngest</u> Bennet, Lydia, will come th Wickham, and the results are far from felicitous will al You Are Very Smart} Though the central characters, Darcy, begin the novel as hostile acquaintances and unlikely friends, they eventually work to understand each selves so that they can marry each other on compatible terms personally, even if their "equal" social status r . When Elizabeth rejects Darcy's first proposal, the argument of only marrying when one is in love is introduc only accepts Darcy's proposal when she is certain she loves him and her feelings are reciprocated. Austen's co g of different marriage ultimately allows readers to question what forms of alliance are desirable, especiall s to privileging economic, sexual, companionate attraction.	n poss roblem wife, iage i rlotte relat ubstan to re- Elizab other emains ed. El mplex y when	ession 欽撞or if s s a cor Lucas, ionship ce (ecc enact w eth and fraug izabet sketch. it cor	o t su p pon wi d em ht h in me
D:\谷歌下载>			
https://bl	og.csdn.	net/vhkjh	wbÇ

xortool.py是基于python的脚本,用于完成一些xor分析,包括:

- 猜想key的长度
- 猜想key的值
- 解密一些经过xoe加密的文件

### 24, can\_has\_stdio?

打开文件是一大串brainfuck密文,在线解码得到flag

: http://tool.bugku.com/brainfuck/?wafcloud=2

### 25, MISCall

不知道是什么文件,用kali看一下,发现是bzip2文件,重命名为 123.bzip2

```
root@kali:~/desktop/share# file 123
123: bzip2 compressed data, block size = 900k
root@kali:~/desktop/share# mv 123 123.bzip2
root@kali:~/desktop/share# ls
123.bzip2 kali.jpg
```

解压:

<pre>root@kali:~/desktop/share# tage</pre>	ar :	xvf	123.bzip2
ctf/			
ctf/flag.txt			
ctf/.git/			
ctf/.git/description			
ctf/.git/refs/			
ctf/.git/refs/heads/			
ctf/.git/refs/heads/master			
ctf/.git/refs/stash		6.46-	Malan and an attached buch
ctf/.git/refs/tags/		nttps	s://blog.csdn.net/vnkjnwbs

发现了一个falg.txt文件和 .git文件夹, flag.txt中没有flag

尝试用 git stash 进行恢复以前修改/删除的文件

查看git 记录,给出了一个最近上传的文件,但这个文件并不存在

```
root@kali:~/desktop/share/ctf# git log
commit bea99b953bef6cc2f98ab59b10822bc42afe5abc (HEAD -> master)
Author: Linus Torvalds <torvalds@klaava.Helsinki.Fi>
Date: Thu Jul 24 21:16:59 2014 +0200
```

Initial commit

查看修改列表,储存列表中有一条记录

```
root@kali:~/desktop/share/ctf# git stash list
stash@{0}: WIP on master: bea99b9 Initial commit
```

校验一下列表中的存储文件



把上面的文件恢复

直接执行 git stash apply 时,会提示文件覆盖自动终止,可以先把flag.txt删除再执行



运行s.py 得到flag:

NCN4dd992213ae6b76f27d7340f0dde1222888df4d3

### 26, 3-1

文件用winhex打开发现是rar文件头,后缀改为rar,解压发现有有个文件名为++\_++的文件,用记事本打开,发现这是一个流量包,后缀改为.pcap,用wireshark打开,

在分组字节流 中搜索字符串 flag,

找到了一个 flag.rar文件,

```
GET /flag.rar HTTP/1.1
User-Agent: Wget/1.14 (linux-gnu)
Accept: */*
Host: 10.10.61:8000
Connection: Keep-Alive
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.6.0
Date: Wed, 02 Aug 2017 02:53:59 GMT
Content-type: application/octet-stream
Content-tength: 169
Last-Modified: Wed, 02 Aug 2017 02:46:46 GMT
Rar!...3..
......v...v.u.
```

文件》导出分组字节流保存为 flag.rar,里面有一个 flag.txt文件

解压发现需要密码,暴力破解无果

会过头来找密码,发现在下一个字节流中(第6个)中,看到了一些管道命令,和一段python加密脚本

💋 Wireshark · 追踪 TCP 流 (tcp.stream eq 6) · qwe.pcap — 🛛	$\times$
	^
Password: jfn	
Last login: Sun Jul 23 10:49:11 from 10.1.10.61 [root@localhost ~]#	
[root@localhost ~]#	
[root@localhost ~]# llss	
[0n.[01;34mctf.[0n flag.txt .[01;34mgit.[0n .[01;34mipc.[0n test.txt . [01;34mthread_syn.[0n anacondarks.cfg .[01;31mflag.rar.[0n flag.txt.1 .[01;34mimage.[0n .[01;34msignal.[0n .[01;34mthread.[0m . [01;34Wtmlle.vim.[0n [root@localhost ~]]# ccdd cc tf/	
[root@localhost ctf]# ccdd ww ireshark/	
[root@localhost wireshark]# llss	
1 2 3 test [root@localhost wireshark]# ccaatt 11	
Rar!3 	
19aaFYsQQKr+hVX6hl2smAUQ5a767TsULEUebWSajEo= <mark>k</mark> root@localhost wireshark]# ppiinngg bbaaiidduuccoomm	
PING baidu.com (111.13.101.208) 56(84) bytes of data. 04 bytes from 111.13.101.208 (111.13.101.208); icmp_seq=1 ttl=48 time=33.4 ms 04 bytes from 111.13.101.208 (111.13.101.208); icmp_seq=2 ttl=48 time=32.1 ms 04 bytes from 111.13.101.208 (111.13.101.208); icmp_seq=3 ttl=48 time=34.7 ms 04 bytes from 111.13.101.208 (111.13.101.208); icmp_seq=4 ttl=48 time=31.9 ms °C baidu.com ping statistics 4 packets transmitted, 4 received, 0% packet loss, time 3037ms rtt min/awg/max/ndw= 31.921/33.067/34.784/1.155 ms [root@localhost wireshark]# ccaatt 33	~
分组 485.107 <u>客户调</u> 分组, 121 服务器 分组, 202 turn(s). 点击选择。	
整个对话(2696 bytes)    显示和保存数据为 ASCII    流	6 🖨
查找:	个(N)
滤掉此流 打印 Save as… 返回 Close Hel	P

💋 Wireshark · 追踪 TCP 流 (tcp.stream eq 6) · qwe.pcap	—		×
from Crypto import Random			^
from Crypto.Cipher import AES			
import sys			
import base64			
IV = 'QWERTYUIOPASDFGH'			
def decrypt(encrypted):			
aes = AES.new(IV, AES.MODE_CBC, IV)			
return aes. decrypt(encrypted)			
<pre>def encrypt(message): .</pre>			
length = 16			
count = len(message)			
padding = length - (count % length)			
nessage = nessage + '(0' * padding			
aes = AES.new(IV, AES.MODE_DEC, IV)			
return aes. encrypt(message)			
te = 'this is a tast'			
example = pronunt(str)			
• • • • • • • • • • • • • • • • • • •			
nrint(decrypt(evam]e))			
分組 643, 107 <u>第戶端</u> 分組, 121 服务额 分組, 202 varn(s). 点齿选择。			~
整个对话(2696 bytes) ~ 显示和保存数据为 ASCII ~		流	6 🗘
查找:	3	査找下一个	(N)
滤掉此流 打印 Save as… 返回	ose	Helj	hwite:

```
from Crypto import Random
from Crypto.Cipher import AES
import sys
import base64
IV = 'QWERTYUIOPASDFGH'
def decrypt(encrypted):
aes = AES.new(IV, AES.MODE_CBC, IV)
return aes.decrypt(encrypted)
def encrypt(message):
length = 16
 count = len(message)
 padding = length - (count % length)
 message = message + '\0' * padding
 aes = AES.new(IV, AES.MODE_CBC, IV)
 return aes.encrypt(message)
str = 'this is a test'
example = encrypt(str)
print(decrypt(example))
```

这是一段加密脚本,密文是19aaFYsQQKr+hVX6hl2smAUQ5a767TsULEUebWSajEo=

```
写出解密脚本,解密得到:
```

```
passwd={No_One_Can_Decrypt_Me}
```

解密 flag.rar得到 flag:

WDCTF{Seclab\_CTF\_2017}

27,适合作为桌面

是一张图片,二话不说先用foremost分离一下,没分离出来东西

放进stegsolve 里看一下,

发现了一张二维码:



扫描得到一段 16进制:

QR Research	- 🗆 X
文件(E) 工具(I) 帮助(H)	
📄 🔝 🔝 🔌 😧	E 💽 🖌
	纠错等级   H(30%) ▼   Auto ▼ 版本   Auto ▼   4   ↓
T	^
	~
内容: 03F30D0A79CB055863000000000000000010000001000000400000 63280200000630000003000001600000430000073 500540600640700643000540900640006400064000060 0D0640E00640900540F000710000700006410007D010078 0007C020083010377D0100715500577C010047486400005 610000009697000000695700000006330000006330000006	Accession of the second s
000007403000000731427494000000666C616774010000 2E707952030000001300000730400000001480106010D 0002800000002800000007304000000312E707974080000 00	0069280000000028000000730400000031 0114014E28010000005203000002800000 003C6D6F64756C653E0100000073000000
解码完成	https://blog.csdm.nei/vhRjhwbs

复制到winhex中,发现了有1.py字符串:

0	1	2	3	4	5	6	7	8	9	A	в	С	D	Е	F	ANSI ASCII
03	F3	0D	0A	79	CB	05	58	63	00	00	00	00	00	00	00	ó yế Xc
00	01	00	00	00	40	00	00	00	73	0D	00	00	00	64	00	@ s d
00	84	00	00	5A	00	00	64	01	00	53	28	02	00	00	00	"ZdS(
63	00	00	00	00	03	00	00	00	16	00	00	00	43	00	00	c C
00	73	78	00	00	00	64	01	00	64	02	00	64	03	00	64	sx d d d d
04	00	64	05	00	64	06	00	64	07	00	64	03	00	64	08	d d d d
00	64	09	00	64	0A	00	64	06	00	64	0B	00	64	0A	00	dddd
64	07	00	64	08	00	64	0C	00	64	0C	00	64	0D	00	64	d d d d d
0E	00	64	09	00	64	0F	00	67	16	00	7D	00	00	64	10	ddg}d
00	7D	01	00	78	1E	00	7C	00	00	44	5D	16	00	7D	02	} x   D] }
00	7C	01	00	74	00	00	7C	02	00	83	01	00	37	7D	01	t   f 7}
00	71	55	00	57	7C	01	00	47	48	64	00	00	53	28	11	qUW  GHd S(
00	00	00	4E	69	66	00	00	00	69	6C	00	00	00	69	61	Nif il ia
00	00	00	69	67	00	00	00	69	7B	00	00	00	69	33	00	ig i{ i3
00	00	69	38	00	00	00	69	35	00	00	00	69	37	00	00	i8 i5 i7
00	69	30	00	00	00	69	32	00	00	00	69	34	00	00	00	i0 i2 i4
69	31	00	00	00	69	65	00	00	00	69	7D	00	00	00	74	il ie i} t
00	00	00	00	28	01	00	00	00	74	03	00	00	00	63	68	(t ch
72	28	03	00	00	00	74	03	00	00	00	73	74	72	74	04	r( t strt
00	00	00	66	6C	61	67	74	01	00	00	00	69	28	00	00	flagt i(
00	00	28	00	00	00	00	73	04	00	00	00	31	2E	70	79	( s 1.py
52	03	00	00	00	01	00	00	00	73	0A	00	00	00	00	01	R s
48	01	06	01	0D	01	14	01	4E	28	01	00	00	00	52	03	H N(R
00	00	00	28	00	00	00	00	28	00	00	00	00	28	00	00	<b>←</b> ← ← ← ← ← ←
00	00	73	04	00	00	00	31	2E	70	79	74	08	00	00	00	s 1.pyt
3C	6D	6F	64	75	6C	65	3E	01	00	00	00	73	00	00	00	<module> s</module>
00																

知道这是一个 pyc文件, 另存为 1.pyc 放进kali中进行反编译, 得到1.py文件:

uncompyle6 1.pyc >1.py

查看1.py文件:



发现是一个函数,用vim在脚本后面加一个 flag()调用这个函数,然后执行这个脚本就得到flag:

flag{38a57032085441e7}

是一张斑马图片,身上是一段条形码,用画图拉长后再在线解码

的得到: FLAG IS TENSHINE

flag: flag{TENSHINE}

# 29,我们的秘密是绿色的

这题还是比较难的,需要好多步才能解出来flag

(1) 首先得知道有 oursecret这个图片解密软件,然后还得找到解密的密码,

密码经过一番尝试后,发现是图片中的绿色的数字连起来0405111218192526,然后得到一个 try.zip文件

Our Sec	ret		Make you	r secrets inv	visible in just 3 ec	sy steps !
	HID			UNH	IDE	
I	Step 1:	Select a carrier file	1	Step1: S	pecify a carrier file	•
		No carrier file select	ed	gr	een.jpg Size: 260	0610 bytes
	Step 2:	Add/remove file or me	essage	Step 2: E	nter password	
	- 🛃 A	dd 🛃 Remove		•••••	•••••	
	Туре	Name	Size (k)	🆄 Ur	hide (dout	ble click to save)
Help				Type File	Name try.zip	Size (k) 1
About	<		>			
Exit	Step 3: Enter pa	Password ssword again		۲		>
C==000 c=========			🖗 Hide			<b>J</b>

(2) 解压try.zip发现需要密码,不过给了点提示:

/JH/	< 1911 E 108 2-3	1743 HAV 174121	6 Pageos	113 22	34010 10345	DT/at-	
$\uparrow$	🌆 try.zip - Z	IP 壓縮檔, 未封裝	达小 689 位	元組			
名稱	^	大小	封裝後	類型	修改的日期	CRC32	你知道coffee的生日是多少么~ <mark>~</mark>
μ.				文件夹			
🚰 flag	J.zip *	592	604	WinRAR ZIP 壓縮相	當 2017/4/20 15:	CC9B3374	
📄 rea	dme.txt *	97	109	文本文档	2017/4/20 14:	BA59D7	

coffee的生日, 别查生日了,直接爆破,纯数字 爆破得到密码: 19950822

(3) 解压后得到 两个文件 flag.zip需要密码,用明文爆破,得到密码: Y29mZmVI

🔏 ARCHPR 4.54 - 0%			-		$\times$				
文件(F) 恢复(R) 帮助(H	)								
	1 (3	1	2						
打开 开始! 停!	上基准则试	升级	帮助	关于	退出				
加密的 ZIP/RAR/ACE/ARJ 5	文件	攻击类型							
C:\Users\HP\Desktop\flag.z	ip	明文			$\sim$				
1									
口令已成功恢复!					$\times$				
Advance	ed Archive Passw	ord Recovery	统计信息:						
总计口令	n,	/a							
总计时间	31	3m 18s 307ms							
平均速度(口令/秒)	n,	/a							
这个文件的口令	Y:	29mZmVl	<b>C</b>						
十六进制口令	5	9 32 39 6d 5a	6d 56 6c		6				
			✔ 确定						
2019/10/22 20:37:47 - 'Y29	mZmVI' 是这个文	(件的一个有刻	次口令		~				
当前口令: n/a 已用时间: 3m 18s 明文攻击正在进行, 尝试抄	洄口令(最长9	平均速度: 剩余时间: 个符号)	n/a 1h 5	4m 9s					
	0	%							
RCHPR version 4.54 (c) 1997	-2012 ElcomSoft	Co. Ltd.	ips://blog.c	schrinel/s	mkjnwbs				

(4) 又得到两个 文件, flag.zip 又需要密码

用 ARCHPR爆破时,提示文件没有加密,那应该是 zip伪加密

将 01 改为 00 保存, 解压得到flag.txt

5	) 4B	03	04	14	00	00	80	08	00	66	76	94	4A	7D	AF	PK	fv".	J}_	
7:	2 9F	1E	00	00	00	1E	00	00	00	08	00	00	00	66	6C	rŸ		fl	
6	L 67	2E	74	78	74	2B	4C	49	29	28	2C	$\mathbf{CF}$	2B	48	2E	ag.txt	+LI)(,Ï-	+H.	
C	3 49	CD	53	2D	28	02	B2	E3	AB	E3	AB	AA	в4	52	1C	ÈIÍS-(	²ã«ã«ª	'R	
D	2 OB	6B	01	50	4B	01	02	3F	00	14	00	01	<mark>0</mark> 9	08	00	ÒkPK	?		
6	5 76	94	4A	7D	AF	72	9F	1E	00	00	00	1E	00	00	00	fv"J}	rŸ		
0	3 00	24	00	00	00	00	00	00	00	20	00	00	00	00	00	Ş			
0	00 (	66	6C	61	67	2E	74	78	74	0A	00	20	00	00	00	flag	.txt		
0	00 (	01	00	18	00	E6	$\mathbf{FC}$	D6	7E	A2	В9	D2	01	2C	E6		æüÖ∼¢¹Ò	,æ	
5	7 65	82	в9	D2	01	2C	E6	57	65	82	в9	D2	01	50	4B	We,¹Ò	,æWe,¹Ò	PK	
0	5 06	00	00	00	00	01	00	01	00	5A	00	00	00	44	00		Z	D	
0	00 (	00	00																

qddpqwnpcplen%prqwn\_{\_zz\*d@gq}

(4) 直接提交,不正确,加flag{}后也不正确,

猜想可能是 栅栏密码 先经过栅栏爆破 得到: qwlr{ddneq\_@dpnwzgpc%nzqqpp\_\*}

(5) 再进行 凯撒爆破 得到flag:

flag{ssctf\_@seclover%coffee\_\*}

### 30, simple\_transfer

得到一个流量包,打开,前面的包好像是在进行端口扫描,全是握手包

直接在分组字节流中搜索字符串flag,找到了一个包,追踪流,没发现有用信息

回过头来,再看,往下倒非红色包的部分,发现有 NFS 协议,觉得有点可疑,

过滤看看

发现 在分组列表中 有 file.pdf

4297 61.844642	10.0.2.5	10.0.2.4	NFS	210 V4 Call (Reply In 4298) GETATTR FH: 0x0163bd75
4298 61.844943	10.0.2.4	10.0.2.5	NFS	266 V4 Reply (Call In 4297) GETATTR
4300 61.845705	10.0.2.5	10.0.2.4	NFS	210 V4 Call (Reply In 4301) GETATTR FH: 0x0163bd75
4301 61.845888	10.0.2.4	10.0.2.5	NFS	266 V4 Reply (Call In 4300) GETATTR
4303 62.583091	10.0.2.5	10.0.2.4	NFS	230 V4 Call (Reply In 4304) LOOKUP DH: 0x0163bd75/file.pdf
4304 62.583488	10.0.2.4	10.0.2.5	NFS	122 V4 Reply (Call In 4303) LOOKUP Status: NFS4ERR_NOENT
4306 62.583621	10.0.2.5	10.0.2.4	NFS	242 V4 Call (Reply In 4307) SETCLIENTID
4307 62.583814	10.0.2.4	10.0.2.5	NFS	130 V4 Reply (Call In 4306) SETCLIENTID
4308 62.583851	10.0.2.5	10.0.2.4	NFS	174 V4 Call (Reply In 4314) SETCLIENTID_CONFIRM
4314 62.584232	10.0.2.4	10.0.2.5	NFS	114 V4 Reply (Call In 4308) SETCLIENTID CONFIRM

导出分组字节流 保存为 file.pdf

但是我打不开 这个破文件,可能是我导出是姿势有问题,

算了直接 用 foremost 分离 流量包,分离得到 了一个pdf文件: (ma的,下次我直接分解)

# HITB{b3d0e380e9c39352c667307d010775ca}

HITB{b3d0e380e9c39352c667307d010775ca} 直接提交

### 31, Just-No-One

这题我也是服了,这没有点耐心还真找不到

下载得到一个 exe文件,那就安装呗,还能咋地,但发现需要password:

🔀 Setup - Binathlon 10	_	
Password This installation is password protected.		
Please provide the password, then dick Next to continue. Password: case-sensitive.	are a	
Password:		
< Back Nex	t >	Cancel

找密码:

尝试使用innounp工具解压压缩包。

innounp的官网地址: http://innounp.sourceforge.net

innounp.exe -x setup.exe

名称 {app} innounp.exe install_script.iss 鍔 setup.exe	修改日期 2018/1/14 15:46 2016/4/11 10:57 2018/1/14 15:46 2018/1/10 9:33	<ul> <li>美型</li> <li>文件夹</li> <li>应用程序</li> <li>ISS 文件</li> <li>应用程序</li> </ul>	大小
<pre>C:\Users\CTF\Desktop\CTF\mis A innounp.exe -x setup.exe ; Version detected: 5500 #1 {app}\bin10.exe Reading slice C:\Users\CTF\D #2 {app}\bin10.c #3 install_script.iss</pre>	c esktop\CTF\misc\s	etup.exe https://blog.csdn.ne	et∕vhkjhwbs

在解压出的install\_script.iss中发现密码信息

; PasswordHash=888e209e5d76a3135d04b8baccabd7936a0dd376

; PasswordSalt=0e559522ecf62077

使用hashcat解密得到密码为:1234567

但实际上并没有什么用,程序安装完打开是这个样子的



提示我们已经看到了flag

最骚的来了,没有任何提示的情况下,能在安装协议里面找到正确 flag 的人是真的大佬,反正我做不来

🖶 Setup - Binathlon 10 — 🛛	$\times$
License Agreement Please read the following important information before continuing.	
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.	
EULA SHALL NOT EXCEED THE PRICE PAID FOR THE SOFTWARE, IF ANY. IN NO EVENT SHALL MORE SMOKED LEET CHICKEN OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, SPECIAL, INCIDENTAL OR INDIRECT DAMAGES OF ANY KIND ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE, EVEN IF MORE SMOKED LEET CHICKEN OR ITS SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR ANY CLAIM BY A THIRD PARTY. 7. RENTAL. YOU MAY NOT LOAN, RENT, OR LEASE THE SOFTWARE. 7A. YOU MAY SUBMIT THIS TO GET TEN POINTS: ILOVERADINGEULAS. 8. UPGRADES. IF THE SOFTWARE IS AN UPGRADE FROM AN EARLIER RELEASE OR RREVIOUSLY RELEASED VERSION, YOU NOW MAY USE THAT UFGRADED PRODUCT ONLY IN ACCORDANCE WITH THIS EULA. IF THE BINATHLON 10 SOFTWARE PRODUCT	~
I accept the agreement     I do not accept the agreement	
< Back Next > C	ancel

谁能想到 这就是flag: 也没个格式, 真想骂这个出题人,

ILOVEREADINGEULAS

### 32, warmup

考察,明文爆破,盲水印

先将 open\_forum.png 压缩为 open\_forum.zip, 然后明文破解(这里我死活破不开,不知道是怎么回事,我醉了)

然后得到,两张图片,用盲水印提取水印: (脚本自己去github里面找)

python bwm.py encode fuli.png fuli2.png res.png



flag{bWm\_Are\_W0nderfu1}

# 33, Erik-Baleog-and-Olaf

是png头,后缀改为.png

用 stegsolve 打开 发现在中间的小人的手中,有一个很小的二维码,不断切换找到一张最清的二维码: 用 截图工具截个图:



直接扫, 扫不出来,

再把这张图放进 stegsolve中 , 找到一张更清的图片, 保存下来



扫描得到flag: flag{#justdiffit}

### 34, Py-Py-Py

只能说花里胡哨,第一次见到 在 pyc/pyo文件中 隐藏信息的 开始的时候把 pyc文件 放进 kali中 反编译为 py文件

uncompyle6 123.pyc > 123.py

然后看到一个加密脚本:

太长了,选出有用的一部分,

```
if __name__ == '__main__':
    while True:
        flag = raw_input('Please input your flag:')
        if flag == crypto(fllag, 'decode'):
            print('Success')
            break
        else:
            continue
```

看到, flag == crypto(fllag,'decode')

直接把这一段代码删除了 添加一句 print crypto(fllag,'decode')

然后就得大 flag 的值 The challenge is Steganography, 直接提交不对, 加flag{}提交仍然不对

```
C:\Python27\python2.exe C:/Users/冊/Desktop/文档/.
The challenge is Steganography
Process finished with exit code 0
```

这句话的意思是 挑战是隐写,

去网上搜了一下,还可以在 pyc文件的字节码中 隐藏 信息的隐写方式,

需要用到 stegosaurus 这个工具, github中有,

python3 stegosaurus.py -x 123.pyc

D:\谷歌下载\stegosaurus-master\stegosaurus-master>python3 stegosaurus.py -x 123.pyc Extracted payload: Flag{HiD3\_Pal0ad\_1n\_Python}

Flag{HiD3\_Pal0ad\_1n\_Python}

### 35, reverse\_it

放进 winhex中发现是 文件 png 的数据进行了 逆置, (这里的逆置是指,字节的逆置)

	96	78	C8	00	00	00	41	00	00	00	20	00	23	10	27	00	-2	٢E	А		#	
	00	00	91	00	00	00	20	00	13	10	00	00	20	00	10	00		<b>`</b>				
	00	00	30	00	82	10	AG	00	00	00	10	00	00	00	50	00		ο,				Ρ
	В1	10	26	00	00	00	10	00	00	00	50	00	A1	10	00	00	±	&		P	1	
	10	00	10	00	00	00	30	00	21	10	70	00	80	00	00	00			0	! p	€	
	A2	00	D4	D4	00	00	66	96	87	54	2D	00	1E	$\mathbf{FF}$	00	00	¢	ôô	f-	-‡T-	Š	Ż
	84	00	84	00	10	10	10	00	64	94	64	Α4	01	00	0E	FF	"	"		d″d	ы	Ÿ
Г	8D	FF											_				Ś	Ż				

数据逆置回来保存为 png, 得到一张逆置的图片:

# SECCON{6in\_tex7}

再把图片逆置过来:

# SECCON{6in\_tex7}

文件逆置:

```
def swap_nibbles(byte):
    return ((byte << 4) | (byte >> 4)) & 0xff

i = open('1234', 'rb')
o = open('1234.txt', 'wb')
o.write(''.join(map(chr, map(swap_nibbles, map(ord, i.read()[::-1])))))
i.close()
o.close()
```

图片翻转:

```
import sys
from PIL import Image
i = Image.open('1234.jpg')
o = Image.new(i.mode, i.size)
idata, odata = i.load(), o.load()
for y in range(i.size[1]):
    for x in range(i.size[0]):
        odata[x, y] = idata[i.size[0] - x - 1, y]
o.save('qwe' + '.jpg')
```

# 36, mysql

提示:我们在Mysql数据库中存放了flag,但是黑客已经把它删除了。你能找回来flag吗?

考点,用 undrop-for-innodb 恢复 mysql被删的数据

undrop-for-innodb 在github上有,

git clone https://github.com/twindb/undrop-for-innodb

cd updrop-for-innodb

make

```
如果提示
make : bison 命令未找到,则需要安装 bison
```

apt install -y bison

然后再执行 make 命令 安装

(这里我也是第一次见这种题,我尽量详细记录一下)

现在确认一下数据恢复的必要条件:一份ibdata1数据文件,一份要恢复的数据库的表结构

在structure.sql中看到了表的结构:

```
CREATE TABLE `user` (
  `id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(10) NOT NULL,
  `password` varchar(32) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

在 mysql文件夹中有 数据空间文件ibdata1文件,现在两个需要的文件都有了

这里我新建了一个 文件夹, backup, 然后把需要用到的几个文件复制进去

c parser, dictionary, stream parser, structure.sql, ibdata1

(1) 解析数据文件

首先,由于mysql将Innodb驱动的数据使用B+tree索引在了数据空间文件ibdata1中,所以需要使用 stream parser工具进行解析:

./stream\_parser -f ibdata1

<pre>root@kali:~/document/undrop-for Opening file: ibdata1 File information:</pre>	-innodb/back LOB -innodb/backu -innodb/backu	kup# <sub>e</sub> ./s p/pages-ib p/pages-ib	trea data	m_parser 1# cd FIL_P 1/FIL_PAGE	-fi AGE_I INDEX	bdatal NDEX/ # ls	
ID of device containing file:	2049						
inode number:	2102140						
protection:000000004.page 0000000	100755	(regula	r fi	ile)			
number of hard links: "/undrop-for	-innodb/backµ						
user ID of owner:	-innodb/backu						
group ID of owner:	LTCENSE						
device ID k(if special file):	Makefil <b>O</b>						
blocksize for filesystem I/O:	pri4096						
number of blocks allocated:	36864						
time of last access:	1571886241	Thu Oct	24	11:04:01	2019	tables_dict.o	
time of last modification:	1571886241	Thu Oct	24	11:04:01	2019	)	
time of last status change:	1571886241	Thu Oct	24	11:04:01	2019	)	
total size, in bytes:	18874368	(18.000	MiE	3)			
Size to process:	18874368	(18.000	MiE	3)		https://blog.oodp.pr	atubkibuba
All workers finished in 0 sec						hups.//biog.csan.ne	www.soverkjnwbs

解析完成后,可以看到同目录下生成一个pages-ibdata1目录,其中包含两个子目录,一个是包含按索引排序的数据页目录,另一个是包含相关类型的数据目录:

系5Ⅰ数据贝日环
安全客 ( bobao.360.cn )

我们下面将主要关注的是第一个子目录即索引好的数据页目录,因为我们要恢复的数据就在里面,其中第一个页文件(00000000000000001.page)里包含所有数据库的表信息和相关的表索引信息,类似一个数据字典,可以使用项目提供的一个脚本recover\_dictionary.sh将其内容放到一个test数据库里详细的查看,这里就不做演示了。

### (2) 解析页文件

既然第一个页文件包含所有数据库表的索引信息,我们就需要先解析它,以模拟mysql查询数据的过程,最终才能找到要恢复的数据。c\_parser工具可以用来解析页文件,不过需要提供该页文件的一个内部结构(表结构)。

项目根目录下有个dictionary目录,里面就包含数据字典用到相关表结构,如用来解析第一个页文件的表结构在 SYS\_TABLES.sql文件

./c\_parser -4Df pages-ibdata1/FIL\_PAGE\_INDEX/0000000000000001.page -t dictionary/SYS\_TABLES.sql | grep ctf

hadrin hadrin	dictionary	- LICENSE	sakila	stros	m narser c	Vadran	+		
<pre>root@kali:~/doc</pre>	ument/undrop-for	-innodb/backup#	./c parser -4Df	pages-ik	odata1/FIL	- PAGE	INDEX/0	0000000	0000000
01.page -t dict	ionary/SYS_TABLE	S.sql   grep ct	f sql parse	ric Strea ril sys p					
000000000506	07000001350221	SYS TABLES to o	"ctf/user"parse	r.o <b>13</b> table	es <b>3</b> lict.c	1	Θ	Θ	
c <mark>⊙</mark> parser.c									
000000000506	07000001350221	SYS TABLES	"ctf/user"	<sup>rse</sup> 13 <sup>test.</sup>	.sla	1	Θ	Θ	
root@kali:~ Pot@kali:~									
SET FOREIGN KEY	CHECKS=0; bdata1								
LOAD DATA LOCAL	/INFILE ///root/d	ocument/undrop-f	or⊢innodb/backu	p/dumps/d	default/SN	/S_TABI	_ES' REP	LACE IN	ΤΟ ΤΑΒ
LE `SYS TABLES `	CHARACTER SET U	TF8 FIELDS TERMI	NATED BY '\t' O	PTIONALLY	( ENCLOSED	) BY ''	'' LINES	STARTI	NG BY
'SYS_TABLES\t'	(`NAME`, `ID`, `	N_COLS`, `TYPE`,	<pre>^ MIX_ID , `MIX</pre>	_LEN`, `(	CLUSTER_NA	ΑME`, `	SPACE`)	;	
STATUS {"rec	ords_expected":	7, "records_dump	ed": 3, "record	s_lost":	true} STA	ATUS EI	١D		
00000000506	07000001350221	SYS_TABLES / page	"ctf/user"	13	3	1	0 https://bloc		vhkihwbs
FOL_PAGE_IN							incpositoiog		ring:1000

该命令使用c parser工具解析数据库表索引信息并过滤出我们想要恢复的有关ctf的文件

我们看到 user这个表的索引值为 13, 通过这个索引值, 再到另外一张表去查询该user表所有的索引信息

该表的结构在"dictionary/SYS\_INDEXES.sql"文件中可以看到,而此表对应的数据页文件是第三个数据页 000000000000000003.page

./c\_parser -4Df pages-ibdata1/FIL\_PAGE\_INDEX/0000000000000003.page -t dictionary/SYS\_INDEXES.sql | grep 13

root@kali:~/doc	ument/undrop-for	-innodb/backup#	./c_pa	rser -4D1	pages-ibdata1	/FIL_PAGE	E_INDEX/0	00000000	0000000
03.paget_dict	07000013E014E	ES salci li grep d	3 10	egl parse	ro tables dict	0 1		0	42
	0/00001350145	STS_INDEXES	13 onarv sh	54 <b>15</b> 98, 36	rser test sh	. · · · <b>1</b>	3	U	42
9490/295	0700001050145	-inpodh#_cd, backup	12	15			-	_	10
000000000506	0/000001350145	SYS_INDEXES# 18	13	15	"PRIMARY"	1	3	Θ	42
94967295parser									
SET FOREIGN KEY	<pre>//CHECKS=0;drop-fo</pre>								
LOAD DATA LOCAL	INFILE '0000000	00506 070000	0135014	5 SYS IN	IDEXES 13	15	"PRIM	IARY"	13
0 4294967	7295								
/root/document/	/undrop-for-innod	b/backup/dumps/	default	/SYS INDE	XES' REPLACE I	NTO TABLE	E`SYS IN	DEXES`	CHARAC
TER SET UTF8 FI	LELDS TERMINATED	BY '\t' OPTIONA	LLY ENC	LOSED BY	'"' LINES STAR	TING BY '	'SYS INDE	XES\t'	(`TABL
E ID', FIIDAGE 'N	AME 'FILN AFIELDS'	SPAC	E`. `PA	GE NO`):					
STATUS {"rec	cords expected":	13, "records du	mped":	3, "recor	ds lost": true	} STATUS	END	g.csdn.net/	vhkjhwbs

这里找到一条user 的索引信息,其在mysql 存储中的索引值为 15,此索引值编号对应的数据页文件中,即存储 了该索引的全部数据 此处我们选择的是主键索引对应的数据页文件进行解析(另外一个索引键应该也可以,只不过方法可能需要有 所区别),终于顺利解析见到了激动人心的数据:

./c\_parser -5f pages-ibdata1/FIL\_PAGE\_INDEX/000000000000015.page -t structure.sql | more

<pre>root@kali:&gt;/document/undrop-for-innodb/backup# ./c_parser -5f pages-ibdata1/FIL PAGE_INDEX/0000000000000000</pre>
5.page -t structure.sql   more 0000000011.page 000000000000000000000000000000000000
SET FOREIGN_KEY_CHECKS=0;
LOAD DATA LOCAL INFILE '/root/document/undrop-for-innodb/backup/dumps/default/user' REPLACE INTO TABLE `u
er` CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' OPTIONALLY ENCLOSED BY '"' LINES STARTING BY 'user\t' (`
d`, `name`{/^`password`);ndrop-for-innodb# ls
STATUS {"records expected": 8, "records dumped": 8, "records lost": false} STATUS END
Page id: 307, Format: COMPACT, Records list: Valid, Expected records: (4 4)
000000000503 84000001340110 user 1 i root" "63a9f0ea7bb98050796b649e85481845"
00000000503 Sec. 8400000134011A suser <sup>CEADM2</sup> d guest sec. 94 e10adc3949ba59abbe56e057f20f883e
000000000503 84000001340124 user 3 "test" "098f6bcd4621d373cade4e832627b4f6"
000000000503 8400000134012E user 4 "flag" "71e55075163d5c6410c0d9eae499c977"
Page id: 307, Found records: 4, Lost records: N0, Leaf page: YES
Page id: 307, Format: COMPACT, Records list: Valid, Expected records: (0 0)
Page id: 307, Found records: 0, Lost records: NO, Leaf page: YES
Page id: 307, Format: COMPACT, Records list: Valid, Expected records: (4 4) https://blog.csdn.net/vhkjhwbs

得到flag:

71e55075163d5c6410c0d9eae499c977

写在最后: (上面的很多题,我也是第一次遇到,也是边刷边学习)

我也是一个刚学习的web安全的小白,也是通过刷题写writeup来提升自己,写的不好的地方勿喷,转身 离开就好,如果觉得有帮助到你,就点个赞就好

