2018巅峰极客writeup(Misc)

原创

Mistsatan ● 于 2018-07-30 10:00:56 发布 ● 1749 ☆ 收藏
 分类专栏: CTF 文章标签: writeup 巅峰极客 CTF Misc 杂项
 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/baidu_28226047/article/details/81238763
 版权



CTF 专栏收录该内容

1 篇文章 0 订阅 订阅专栏 原文地址: https://mistsatan.github.io/articles/2018-Peakgeek-Writeup-Misc.html

作为一个渣渣,带着去看各路神仙打架的想法报名了这次的"巅峰极客"CTF比赛,7.21第一场线上赛果然从头水到尾。。。 为了方便之后巩固,把近几天学习大佬们的writeup过程中总结整理的内容记录下来,第一次写成博客。

我只是个搬运工,以下是众大佬们的writeup:

- 白帽100
- 双螺旋
- 无糖
- 还有一个
- 最后一个

Misc

1.warmup - 100pt

首先拿到一张图片warmup.bmp,用Stegsolve打开:

java -jar Stegsolve.jar



分别查看各个通道的LSB,发现了存放的数据是用Ook!和brainfuck加密的:

		C.S.C.		Calor.
Extract Preview		Extract Preview		Extract Preview
2e2e2e2e2e202e2e 2e2e2e202e2e2e2e	4f6f6b2e204f6f6b 2e204f6f6b2e204f	Ook. Ook . Ook. O	2b2b2b2b2b2b202b5b 2d3e2b202b2b2b2b2b	+++++ +[->+ ++++
2e202e2e2e2e2e2e2 213f21212e203f2e !?!!. ?.	6f6b2e204f6f6b2e 204f6f6b2e204f6f	ok. Ook. Ook. Oo	2b203c5d3e2b2b20 2b2b2b2b2b2b2b2b2b2b	+ <]>++ +++++ ++
2e2e2e202e2e2e2e 2e202e2e2e2e2e2e2	6b2e204f6f6b2e20 4f6f6b2e204f6f6b	k. Ook. Ook. Ook	2e3c2b202b2b5b2d 3e202b2b2b3c5d20	.<+ ++[- > +++<]
2e2e2e2e2e2e202e3f 2e3f21202e3f2e2e? .?! .?	2e204f6f6b2e204f 6f6b2e204f6f6b2e	. Ook. O ok. Ook.	3e2b2e2d2d202e3c 2b2b2b2b2b2b2b5b	>+< +++ +++[
2e202e212e2e2e0a 2e2e2e2e2e2e2e2e!	204f6f6b2e204f6f 6b210a4f6f6b3f20	Ook. Oo k!.Ook?	2d203e2b2b2b2b0a 2b2b3c5d3e202b2b	- >++++. ++<]> ++
2e2e2e20212e3f2e 2e202e2e2e2e2e2e20 !.?	4f6f6b21204f6f6b 21204f6f6b2e204f	Ook! Ook ! Ook. O	2b2b2b2b2b2b2b2b2e 3c202b2b2b2b2b2b2b2	+++ +++. < +++++
213f21212e203f21 2121212021213f2e !?!!. ?! !!! !!?.	616b312041616b2e 2041616b2e204161	ok? Ook. Ook. Oo	2b5b2d3e2d202d2d 2d2d2d203c5d3e2d	+[-> <]>-
3f20212e3f212120 2121212e2e202e2e ? !.?!! !!!	6D2e2U41616D2e2U 41616D2e2U41616D	k. Ook. Ook. Ook		
2e2e2e202e2e2e2e2e2e2e202e212e3f2e0a!.?	2046666522041 616D262041616D26	. ook. o ok. ook.		<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
2e2e2e2e2e2e2e2e2e2e2e2e2e2e2e2e2e2e2e	204161602204161 6022204161602220	00k00 k. 00k.		·····
Dit Dianos Ord	Bit Planes	Order	- Bit Planes	Orde
Dit Plates	Dir Hanoo	01401	Dit i Miloo	or do
Alpha 7 6 5 4 3 2 1 0 Ex	Alpha 7 6 5 4 3	2 1 0 Extr	Alpha 7 6 5 4 3	2 1 0 Extr
Red 7 6 5 4 3 2 1 2 0 Bi	Red 7 6 5 4 3	2 1 0 Bit C	Red 7 6 5 4 3	2 1 0 Bit C
Green 7 6 5 4 3 2 1 0		Bit	Green 7 6 5 4 5	Bit
				(
Draview Settings	Preview Settings		Preview Settings	
Include Hex Dump In Dravious	Include Hex Dump In Preview		Include Hex Dump In Preview	(
Include Hex Dump in Preview	include flex builtp in Preview	P.	include tiex builtp in Freview	
Draviour Court Frid	Provioue	Save Text Save Din	Droviour	Save Text Save Din
Preview Save rext Save Bin	Pleview	JAVE TEAL JAVE DIII	https://blog	Save Jent

分别将三块数据保存出来,用这个网站在线解密,得到以下三部分内容,拼起来就是flag:

- flag{db640436-
- 7839-4050-8339
- -75a972fc553c

flag{db640436-7839-4050-8339-75a972fc553c}

2.loli - 150pt

拿到一张图片1.png,看起来像二维码,用工具扫描,得到一个提示255:



	*
已解码数据 1:	*
位置:(10.0,2.0)-(1072.0,2.0)-(10.0,1064.0)-(1072.0,1064.0) 颜色正常,正像 版本:40 纠错等级:H,掩码:7 内容: 虽然你很幸苦,但是这不是flag~ flps:255	4
解码完成 https://blog.csdn.net/baidu_282260	47/

根据另一个提示0xFF,将1.png放到WinHex中,0xFF异或整个文件,在得到的文件末尾看到字符串"black and white":

Offset	0	1	2	3	4	- 5	6	7	8	9	Α	В	С	D	E	F		
0034F850	7F	BB	33	58	FE	33	44	33	90	2F	63	18	E1	30	77	A3	»3Xþ3D3 /c á0w£	
0034F860	05	25	FA	88	B8	40	AF	72	18	7D	0A	1A	46	79	F3	87	%ú^,0 [−] r } Fyó‡	
0034F870	C8	1B	D6	F2	8B	4B	74	41	37	в0	3B	56	B 3	F1	B9	48	È Öò <kta7°;v³ñ³h< td=""><td></td></kta7°;v³ñ³h<>	
0034F880	7C	16	02	94	BE	D2	66	1A	DO	A1	Α2	51	C6	F2	C9	9B	″¾Òf Ð;¢QÆòÉ>	
0034F890	ЗF	C4	2B	6B	Α4	0C	8C	22	E1	9D	2F	2B	A2	54	F5	E5	?Ä+k¤ Œ"á /+¢Tõå	
0034F8A0	F2	4B	DF	56	65	09	F2	ЗA	D6	A3	59	57	20	7C	37	04	òKßVe ò:Ö£YW ∣7	
0034F8B0	1B	98	20	E2	B9	6E	01	57	B1	4F	F6	49	AD	E8	E5	3C	~ â¹n W±OöI-èå≺	
0034F8C0	1F	34	A 8	E6	43	07	A 8	C7	CE	29	56	66	OF	34	B2	66	4″æC ″ÇÎ)Vf 4⁵f	
0034F8D0	D1	AE	86	EF	86	30	AA	72	FB	Α9	B7	BF	5A	31	24	D2	Ñ©†ï†0ªrû© ¿Z1\$Ò	
0034F8E0	E6	93	37	7F	E8	2F	8E	EE	ЗF	EE	42	E3	6D	37	00	00	æ"7 è/Žî?îBãm7	
0034F8F0	00	00	49	45	4E	44	AE	42	60	82	00	00	00	00	00	00	IEND@B`,	
0034F900	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0034F910	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0034F920	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0034F930	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0034F940	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0034F950	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0034F960	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0034F970	00	00	00	00	62	6C	61	63	6B	20	61	6E	64	20	77	68	black and wh	
0034F980	69	74	65					h	ttn	s · .	/ /h	10	σ. c	has	n. i	het	/ ite i du 2822604	7
								11	p	U + /		10	B. C	u		10.0		

用binwalk查看得到的文件,发现其中包含了一个图片:



根据隐藏图片的偏移,用WinHex将图片提取出来,能够看到,图片中,按照行来看,以8个像素点(黑/白)为一组,每一组之间用白点来分隔,根据前面"black and white"提示,不难联想到应该是二进制流转换成字符:





参考各大佬的writeup,写出脚本如下:

```
from PIL import Image
```

```
def getBinaryToChar():
   ans = ""
   img = Image.open('./noname.png').convert('L')
   # 遍历像素点 (按行读)
   width, height = img.size
   for h in range(height - 1): # 是否-1均可,最后一行是11个011111111
       for w in range(width - 1): # 需要-1,最后一列全是0
           pixel = img.getpixel((w, h))
           if pixel == 0:
               color = 1 # 黑点置1
               color = 0 # 白点置0
           count += 1
           ans += str(color)
                   binStr += chr(int(ans,2))
               ans = ""
   return binStr
if ___name___ == '___main___':
   strr = getBinaryToChar()
   print strr
   with open('res.txt', 'wb') as f:
       f.write(strr)
```

打开得到的res.txt,看到flag:



flag{e0754197-e3ab-4d0d-b98f-96174c378a34}

3.flows - 200pt

拿到一个pcap包,用wireshark打开,发现是USB协议。 按照协议排序,一个个点击查看包内容,发现有个包(95)里面疑似有tips:

	95 11.4816	500 hos	t 3.	2	USB	539 URB_BULK out
	98 11.4972	200 hos	t 3.	2	USB	4123 URB_BULK out
	111 11.4972	200 hos	t 3.	2	USB	539 URB_BULK out
1			III			
Đ	Frame 95:	539 bytes or	n wire (4312 bits),	539 bytes captu	red (4312 bits)	
Ð	USB URB					
	Leftover Ca	apture Data:	: 746970733a0a0a31e	38081e994aee79b98	8e79a847573622e	

L	0000	1b	00	10	8b	bc	06	80	fa	ff	ff	00	00	00	00	09	00		
L	0010	00	02	00	03	00	02	03	00	02	00	00	74	69	70	73	3a		tips:
L	0020	0a	0a	31	e3	80	81	e9	94	ae	e7	9b	98	e7	9a	84	75		u
L	0030	73	62	2e	63	61	70	64	61	74	61	e7	ac	ac	e4	b8	80	sb.capda	ta
L	0040	e5	ad	97	e8	8a	82	e4	b8	ba	30	78	30	32	e7	9a	84		.0x02
L	0050	e4	bb	a3	e8	a1	a8	e6	8c	89	e4	ba	86	73	68	69	66		shif
L	0060	74	e9	94	ae	0a	32	e3	80	81	e9	bc	a0	e6	a0	87	e7	t2	
L	0070	9a	84	75	73	62	2e	63	61	70	64	61	74	61	e5	8f	aa	usb.ca	pdata
L	0080	e7	94	a8	e5	85	b3	e5	bf	83	e7	ac	ac	e4	b8	80	e5		and not /haidu 20226047
L	0090	ad	97	e8	8a	82	00	00	00	00	00	00	00	00	00	00	00 5	.//DIUg.	csun. net/baluu_20220047
	00-00	00	00	00	00	$\mathbf{n}\mathbf{n}$	00	00	00	00	00	00	$\mathbf{n}\mathbf{n}$	00	00	$\mathbf{n}\mathbf{n}$	00		

将数据另存到txt中,打开可以看到两个提示:

🗐 95-tips.txt - 记事本	1.00	
文件(E) 编辑(E) 格式(O) 查看	(⊻) 帮助(<u>H</u>)	
kips:1、键盘的usb.capdata第 只用关心第一字节	一字节为OxO2的代表按了shift键2、	鼠标的usb.capdata 🔺
	https://blog.csdn.net/	/baidu_28226047

继续查看包,发现两个长度比较大的包很可疑(55和74):

No.	Time	Source	Destination	Protocol	<mark>Length •</mark> Info
55	11.294400	host	3.2	USB	11803 URB_BULK out
74	11.466000	host	3.2	USB	10267 URB_BULK out
89	11.481600	host	3.2	USB	8219 URB_BULK out
68	3 11.466000	host	3.2	USB	8219 URB_BULK out
49	11.278800	host	3.2 https://h	USB. c.s.dn. ne	+/ 8219 URB_BULK out
98	3 11.497200	host	3.2	USB	4123 URB_BULK out

分别将数据另存出来,用010打开,看到这两个文件头都是D4C3B2A1,就是pcap文件:

	0	1	2	3	4	5	6	7	8	9	A	B	ç	D	E	F	0123456789ABCDEF
0000h:	D4	C3	B2	A1	02	00	04	00	00	00	00	00	00	00	00	00	Ôú;
0010h:	FF	FF	00	00	F9	00	00	00	9D	ЗF	4F	5B	A 8	04	03	00	ÿÿù?0["
0020h:	25	00	00	00	25	00	00	00	1B	00	10	10	DA	05	80	FA	188Ú.€ú
0030h:	FF	FF	00	00	00	00	09	00	01	01	00	01	00	81	01	0A	YY

修改后缀为.pcap,用wireshark打开,有效数据存放在leftover capture data这里:

No.	Time	Source	Destination	Protocol	Length Info		
1	0.000000	3.1	host	USB	35 URB_INTE	RRUPT	in
2	0.093600	3.1	host	USB	35 URB_INTE	RRUPT	in
3	0.483600	3.1	host	USB	35 URB_INTE	RRUPT	in
4	0.592800	3.1	host	USB	35 URB_INTE	RRUPT	in
5	0.702000	3.1	host	USB	35 URB_INTE	RRUPT	in
6	0.826800	3.1	host	USB	35 URB_INTE	RRUPT	in
7	1.092000	3.1	host 🖉	100	×	RUPT	in
8	1.185600	3.1	host 🦰	100		RUPT	in
9	1.606800	3.1	host			RUPT	in
10	1.700400	3.1	host 🦳	The capture file appear	rs to have been cut	RUPT	in
11	1.840800	3.1	host	short in the middle of a	a packet.	RUPT	in
12	1.965600	3.1	host			RUPT	in
13	2.277600	3.1	host			RUPT	in
14	2.402400	3.1	host			RUPT	in
15	2.605200	3.1	host		UK	RUPT	in
16	2.714400	3.1	host				in
17	3.182400	3.1	host	USB	35 URB_INTE	RRUPT	in
18	3.276000	3.1	host	USB	35 URB_INTE	RRUPT	in
19	3.588000	3.1	host	USB	35 URB_INTE	RRUPT	in
20	3.681600	3.1	host	USB	35 URB_INTE	RRUPT	in
21	3.790800	3.1	host	USB	35 URB_INTE	RRUPT	in
22	3.900000	3.1	host	USB	35 URB_INTE	RRUPT	in
23	4.118400	3.1	host	USB	35 URB_INTE	RRUPT	in
•							
🗄 Fra	me 1: 35 bytes	on wire (280 bits),	35 bytes captu	red (280 bits)			
🕀 USE	URB						
Lef	tover Capture	Data: 01000000000000	000				

tshark -r pack55.pcap -T fields -e usb.capdata > pack55.txt tshark -r pack74.pcap -T fields -e usb.capdata > pack74.txt



接下来就需要从txt文件中过滤出键盘击键和鼠标相关的流量: 【这里主要参考这位大佬的博客以及这个博客】

• 键盘数据包的数据长度为8个字节。每次key stroke都会产生一个keyboard event usb packet(所以第一个tips所说第一个字 节为02表示按下了Left Shift键)

Byte 1

|-bit0: Left Control是否按下,按下为1
|-bit1: Left Shift 是否按下,按下为1
|-bit2: Left Alt 是否按下,按下为1
|-bit3: Left GUI 是否按下,按下为1
|-bit4: Right Control是否按下,按下为1
|-bit5: Right Shift 是否按下,按下为1
|-bit6: Right Alt 是否按下,按下为1
|-bit7: Right GUI 是否按下,按下为1
Byte2 暂不清楚,有的地方说是保留位
Byte3-Byte8 这六个为普通按键,击键信息集中在第3个字节

鼠标数据包的数据长度为4个字节。鼠标移动时表现为连续性,与键盘击键的离散性不一样,不过实际上鼠标动作所产生的数据包也是离散的

Byte1 代表按键,当取0x00时,代表没有按键、为0x01时,代表按左键,为0x02时,代表当前按键为右键 **Byte2** 可以看成是一个signed byte类型,其最高位为符号位,当这个值为正时,代表鼠标水平右移多少像素,为负时,代表水平左移多少 像素

Byte3 与第二字节类似,代表垂直上下移动的偏移

第一个包pack55.txt为键盘数据包,需要按照对应关系将键盘按键输出出来,根据第一个tips,注意第一个字节为02表示按了shift 键,在大佬的脚本基础上稍作修改,脚本如下:

```
0x09:"f", 0x0A:"g", 0x0B:"h", 0x0C:"i", 0x0D:"j",
           0x0E:"k", 0x0F:"l", 0x10:"m", 0x11:"n", 0x12:"o",
           0x18:"u", 0x19:"v", 0x1A:"w", 0x1B:"x", 0x1C:"y",
           0x1D:"z", 0x1E:"1", 0x1F:"2", 0x20:"3", 0x21:"4",
           0x22:"5", 0x23:"6", 0x24:"7", 0x25:"8", 0x26:"9",
           0x27:"0", 0x28:"\n",0x29:"[ESC]", 0x2A:"\b", 0X2B:"\t",
           0x31:"\\",0x32:"`", 0x33:";", 0x34:"'", 0x36:",",
           0x37:".", 0x38:"/" }
0x09:"F", 0x0A:"G", 0x0B:"H", 0x0C:"I", 0x0D:"J",
                0x0E:"K", 0x0F:"L", 0x10:"M", 0x11:"N", 0x12:"O",
                0x13:"P", 0x14:"Q", 0x15:"R", 0x16:"S", 0x17:"T",
                0x18:"U", 0x19:"V", 0x1A:"W", 0x1B:"X", 0x1C:"Y",
                0x1D:"Z", 0x1E:"!", 0x1F:"@", 0x20:"#", 0x21:"$",
                0x27:")", 0x28:"\r",0x29:"[ESC]", 0x2A:"\b", 0X2B:"\t",
                0x37:">", 0x38:"?" }
def keyboard extract():
   output = ""
   keys = open('pack55.txt')
   for line in keys:
      if len(line)!= 24:
          continue
      list = line.split(":")
          continue
      num = int(list[2], 16)
       if num in mappings:
              output += mappings_shift[num]
             output += mappings[num]
          output += '[unknown]'
   keys.close()
   print output
if ___name___ == '___main__':
   keyboard_extract()
```

第二个包pack74.txt为鼠标数据包,按照第二个tips,只关注第一个字节,猜测01表示0,02表示1,将其提取出来,脚本如下:

```
def bin2str(bin):
    str = ''
    mo = len(bin)%8
    if (mo):
        bin= '0'*(8-mo) + bin
    chars = re.findall(r'.{8}',bin)
    for char in chars:
        str += chr(int(char, 2))
    return str
def mouse_extract():
    binstr = ""
    keys = open('pack74.txt')
    for line in keys:
        list = line.split(":")
        if list[0]=='00':
            continue
        elif list[0]=='00':
            continue
        elif list[0]=='01':
            binstr += "0"
        elif list[0]=='02':
            binstr += "1"
        str = bin2str(binstr)
        keys.close()
        print str
if __name__ == '__main__':
        mouse_extract()
```

分别运行以上两个脚本,得到两个字符串,拼起来就是flag了:

- flag{u5b_key
- bo4rd_m0use}

flag{u5b_keybo4rd_m0use}