

网刃杯复现misc+流量包

原创

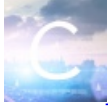
@Demo 于 2021-09-14 09:55:23 发布 222 收藏

分类专栏: [misc](#) 文章标签: [c#](#) [自动驾驶](#) [自然语言处理](#)

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/qq_47886905/article/details/120272377

版权



[misc](#) 专栏收录该内容

9 篇文章 0 订阅

订阅专栏

签到

cipher.txt

```
U2FsdGVkX1+WTSHuJcCjvHj/gcwL0C7u37XtW4idGcpci3H913I=
```

U2F那些, 考虑为aes,des啥的

发现flag.txt是零宽度字符,

Text in Text Steganography Sample

Original Text: (length: 47)

welcome to wang ren, hope you have a good time!

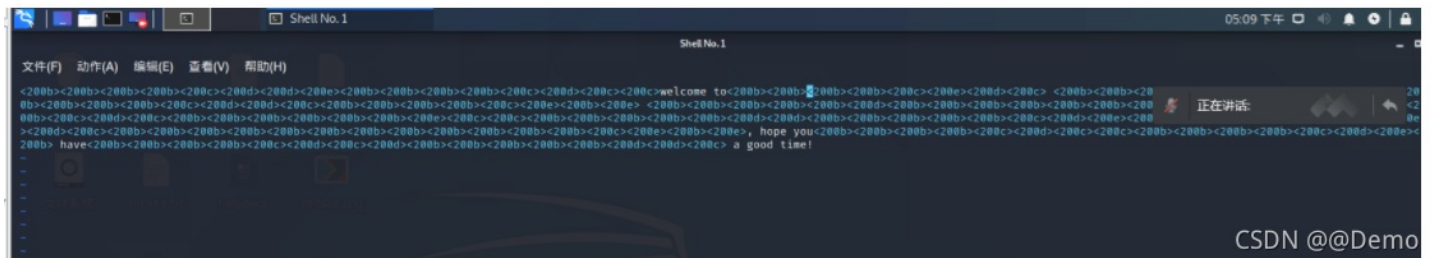
Hidden Text: (length: 19)

key is md5(my/Elself)

Steganography Text: (length: 199)

welcome to wang ren, hope you have a good time!

CSDN @@Demo



md5加密一下文件

```
f71b6b842d2f0760c3ef74911ffc7fdb
```

测试发现是rabbit, 解密获得flag

flag(WelY0me_2_bOI3an)

U2FsdGVkX1+WTSHujcCjvHj/gcwL0C7u37XtW4idGpci3H913l=

密码: Rabbit 加密 解密 清空

CSDN @@Demo

mspaint

```
root@kali:~/volatility_2.6_lin64_standalone# volatility -f mspaint.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO      : volatility.debug      : Determining profile based on KDBG search...
           Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win
Win7SP1x64_23418
           AS Layer1           : WindowsAMD64PagedMemory (Kernel AS)
           AS Layer2           : FileAddressSpace (/root/volatility_2.6_lin64_standalone/mspaint.raw)
           PAE type            : No PAE
           DTB                  : 0x187000L
           KDBG                 : 0xf8000403c0a0L
           Number of Processors : 1
           Image Type (Service Pack) : 1
           KPCR for CPU 0       : 0xfffff8000403dd00L
           KUSER_SHARED_DATA     : 0xfffff78000000000L
           Image date and time  : 2021-09-08 09:47:28 UTC+0000
           Image local date and time : 2021-09-08 17:47:28 +0800
```

CSDN @@Demo

看一下iehistory

```
File Offset: 0x100, Data Offset: 0x0, Data Length: 0xc4
*****
Process: 2440 explorer.exe
Cache type "URL " at 0x3606d00
Record length: 0x100
Location: Visited: qiyue@https://pan.baidu.com/s/1ZobcXkTaKm09R_nTlOno2w
Last modified: 2021-09-08 09:45:55 UTC+0000
Last accessed: 2021-09-08 09:45:55 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0xa8
*****
Process: 2440 explorer.exe
Cache type "URL " at 0x3606e00
Record length: 0x180
Location: Visited: qiyue@https://pan.baidu.com/share/init?surl=ZobcXkTaKm09R_nTlOno2w
Last modified: 2021-09-08 09:46:13 UTC+0000
Last accessed: 2021-09-08 09:46:13 UTC+0000
File Offset: 0x180, Data Offset: 0x0, Data Length: 0xb4
*****
Process: 2440 explorer.exe
Cache type "URL " at 0x3606f80
Record length: 0x100
Location: Visited: qiyue@file:///C:/Program%20Files/Common%20Files/key.png
Last modified: 2021-09-08 09:47:01 UTC+0000
Last accessed: 2021-09-08 09:47:01 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0xac
*****
```

CSDN @@Demo

发现key.png和百度网盘

```
*****
Process: 3568 iexplore.exe
Cache type "URL " at 0x3db5000
Record length: 0x100
Location: :2021090820210909: qiyue@https://pan.baidu.com/share/init?surl=ZobcXkTaKm09R_nTl0no2w
Last modified: 2021-09-08 17:46:13 UTC+0000
Last accessed: 2021-09-08 09:46:13 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x0
*****
Process: 3568 iexplore.exe
Cache type "URL " at 0x3db5100
Record length: 0x100
Location: :2021090820210909: qiyue@Host: pan.baidu.com
Last modified: 2021-09-08 17:45:58 UTC+0000
Last accessed: 2021-09-08 09:45:58 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x0
*****
Process: 3568 iexplore.exe
Cache type "URL " at 0x3db5200
Record length: 0x100
Location: :2021090820210909: qiyue@https://pan.baidu.com/share/init?surl=ZobcXkTaKm09R_nTl0no2w
Last modified: 2021-09-08 17:46:12 UTC+0000
Last accessed: 2021-09-08 09:46:12 UTC+0000
File Offset: 0x100, Data Offset: 0x0, Data Length: 0x0
CSDN @@Demo
```

```
volatility -f mspaint.raw --profile=Win7SP1x64 filescan | grep key.png
```

```
Volatility Foundation Volatility Framework 2.6
0x000000003e96e7d0 2 0 -W-rwd \Device\HarddiskVolume2\Program Files\Common Files\key.pngare-qiyue\VMwareDnD\20566876\key.png
```

```
volatility -f mspaint.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000003e96e7d0 -D ./
```



得到百度网盘的密码

cipher.zip

领红包 保存到网盘

下载(7M)

简易下载助手

保存到手机

举报

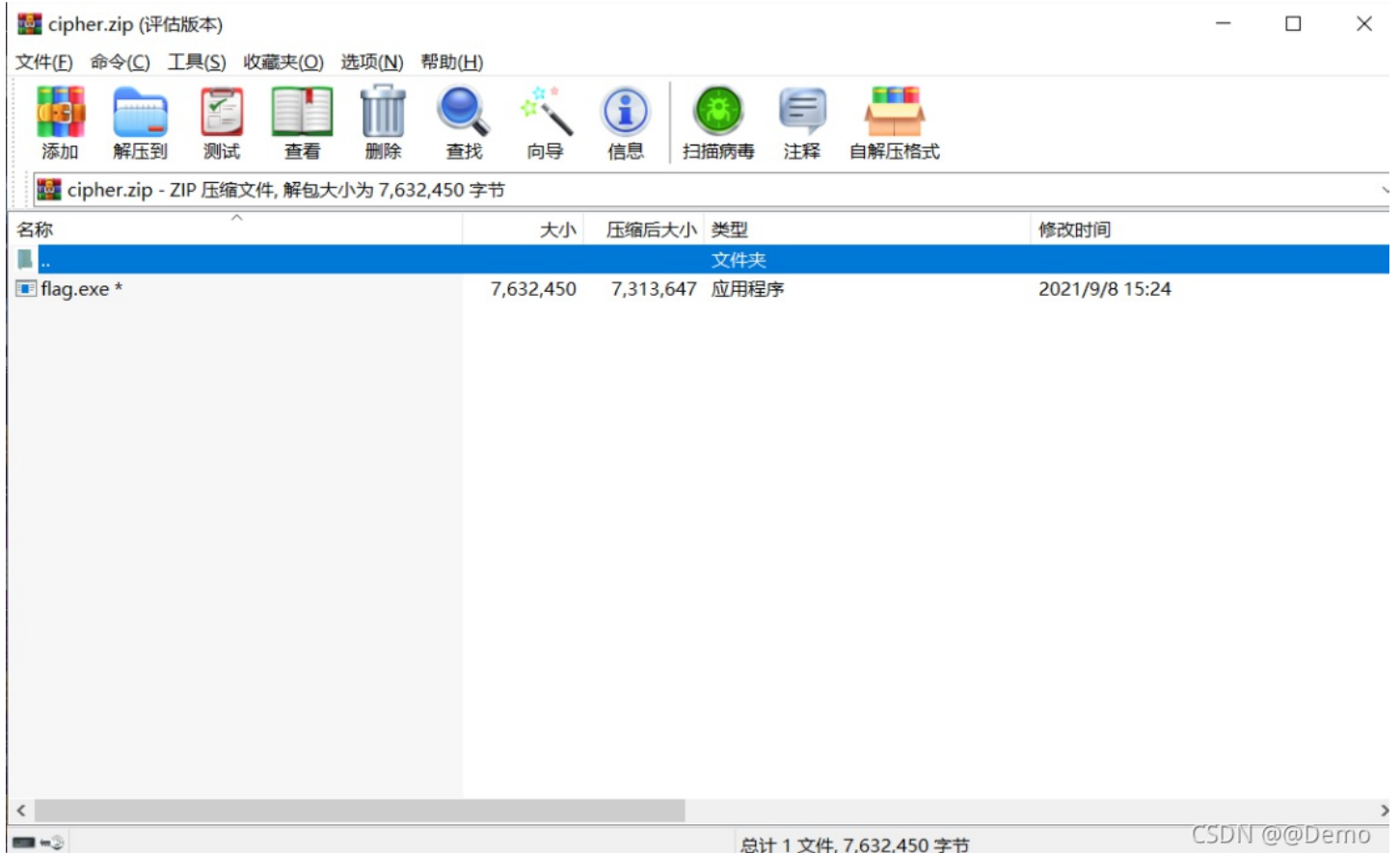
2021-09-08 17:44 过期时间: 永久有效



文件大小:7M

打开压缩包

CSDN @@Demo



发现加密的压缩包，还需要key

```
0060 csrss.exe 392 376 10 370 1 0 2021-01-29 15:43:49 UTC+0000
c060 winlogon.exe 428 376 5 118 1 0 2021-01-29 15:43:49 UTC+0000
e530 services.exe 484 384 8 218 0 0 2021-01-29 15:43:49 UTC+0000
cb30 lsass.exe 500 384 8 758 0 0 2021-01-29 15:43:49 UTC+0000
```


0b30	lsm.exe	508	384	11	145	0	0	2021-01-29	15:43:49	UTC+0000
d670	svchost.exe	616	484	12	371	0	0	2021-01-29	15:43:50	UTC+0000
4060	vmacthlp.exe	676	484	3	53	0	0	2021-01-29	15:43:50	UTC+0000
5b30	svchost.exe	720	484	10	322	0	0	2021-01-29	15:43:50	UTC+0000
c890	svchost.exe	796	484	21	593	0	0	2021-01-29	15:43:50	UTC+0000
5890	svchost.exe	856	484	23	464	0	0	2021-01-29	15:43:50	UTC+0000
f750	svchost.exe	884	484	31	1016	0	0	2021-01-29	15:43:50	UTC+0000
f5f0	svchost.exe	272	484	15	428	0	0	2021-01-29	15:43:51	UTC+0000
fb30	svchost.exe	788	484	18	506	0	0	2021-01-29	15:43:51	UTC+0000
c220	spoolsv.exe	1156	484	13	278	0	0	2021-01-29	15:43:51	UTC+0000
a830	svchost.exe	1184	484	19	325	0	0	2021-01-29	15:43:51	UTC+0000
a060	svchost.exe	1280	484	20	307	0	0	2021-01-29	15:43:52	UTC+0000
db30	VGAuthService.	1340	484	3	86	0	0	2021-01-29	15:43:52	UTC+0000
c060	vmtoolsd.exe	1456	484	9	267	0	0	2021-01-29	15:43:52	UTC+0000
bb30	svchost.exe	1632	484	6	93	0	0	2021-01-29	15:43:52	UTC+0000
d960	msdtc.exe	1944	484	12	144	0	0	2021-01-29	15:43:53	UTC+0000
cb30	WmiPrvSE.exe	1332	616	10	212	0	0	2021-01-29	15:43:54	UTC+0000
9830	taskhost.exe	2320	484	9	218	1	0	2021-01-29	15:45:19	UTC+0000
9730	dwm.exe	2400	856	5	236	1	0	2021-01-29	15:45:19	UTC+0000
5b30	explorer.exe	2440	2392	46	1207	0	0	2021-01-29	15:45:19	UTC+0000
e530	vmtoolsd.exe	2584	2440	9	223	1	0	2021-01-29	15:45:20	UTC+0000
f060	SearchIndexer.	2788	484	14	791	0	0	2021-01-29	15:45:26	UTC+0000
fb30	wmpnetwk.exe	2916	484	15	465	0	0	2021-01-29	15:45:26	UTC+0000
f060	svchost.exe	1444	484	4	266	0	0	2021-01-29	15:45:27	UTC+0000
5b30	sppsvc.exe	1404	484	4	152	0	0	2021-01-29	15:45:52	UTC+0000
eb30	svchost.exe	2892	484	14	324	0	0	2021-01-29	15:45:53	UTC+0000
24f0	cmd.exe	3316	2440	1	19	1	0	2021-09-08	07:35:47	UTC+0000
7060	conhost.exe	2684	392	2	59	1	0	2021-09-08	07:35:47	UTC+0000
3b30	mspaint.exe	2296	2440	5	119	1	0	2021-09-08	07:37:10	UTC+0000
f060	svchost.exe	2196	484	6	103	0	0	2021-09-08	07:37:10	UTC+0000
9b30	dllhost.exe	1524	616	11	179	1	0	2021-09-08	09:36:37	UTC+0000
3920	audiodg.exe	1824	796	5	132	0	0	2021-09-08	09:45:11	UTC+0000
55b0	iexplore.exe	2616	2440	16	607	1	1	2021-09-08	09:45:41	UTC+0000
5b30	iexplore.exe	3916	2616	25	682	1	1	2021-09-08	09:45:41	UTC+0000
5930	iexplore.exe	3568	2616	22	623	1	1	2021-09-08	09:45:51	UTC+0000
d190	SearchProtocol	1560	2788	8	340	0	0	2021-09-08	09:45:52	UTC+0000
7060	SearchFilterHo	3288	2788	6	140	0	0	2021-09-08	09:45:52	UTC+0000
7340	DumpIt.exe	1064	2440	1	25	1	1	2021-09-08	09:47:25	UTC+0000
0060	conhost.exe	1856	392	2	58	1	0	2021-09-08	09:47:25	UTC+0000
3920	dllhost.exe	3272	616	6	91	1	0	2021-09-08	09:47:25	UTC+0000

CSDN@@Demo

```

volatility_2.6_lin64_standalone# volatility -f mspaint.raw --profile=Win7SP1x64 cmdscan
Foundation Volatility Framework 2.6
*****
s: conhost.exe Pid: 2684
y: 0x1bfde0 Application: cmd.exe Flags: Allocated, Reset
3 LastAdded: 2 LastDisplayed: 2
0 CommandCountMax: 50
0x5c
f430: I am a painter
0f70: My favorite thing is painting!!!
2c00: I usually have a good habit of saving screenshots
061c0:
0158:
*****
s: conhost.exe Pid: 1856
y: 0x2afde0 Application: DumpIt.exe Flags: Allocated
0 LastAdded: -1 LastDisplayed: -1
0 CommandCountMax: 50
0x5c

```

CSDN @@Demo

想到截屏

```
volatility -f mspaint.raw --profile=Win7SP1x64 screenshot -D ./
```



CSDN @@Demo

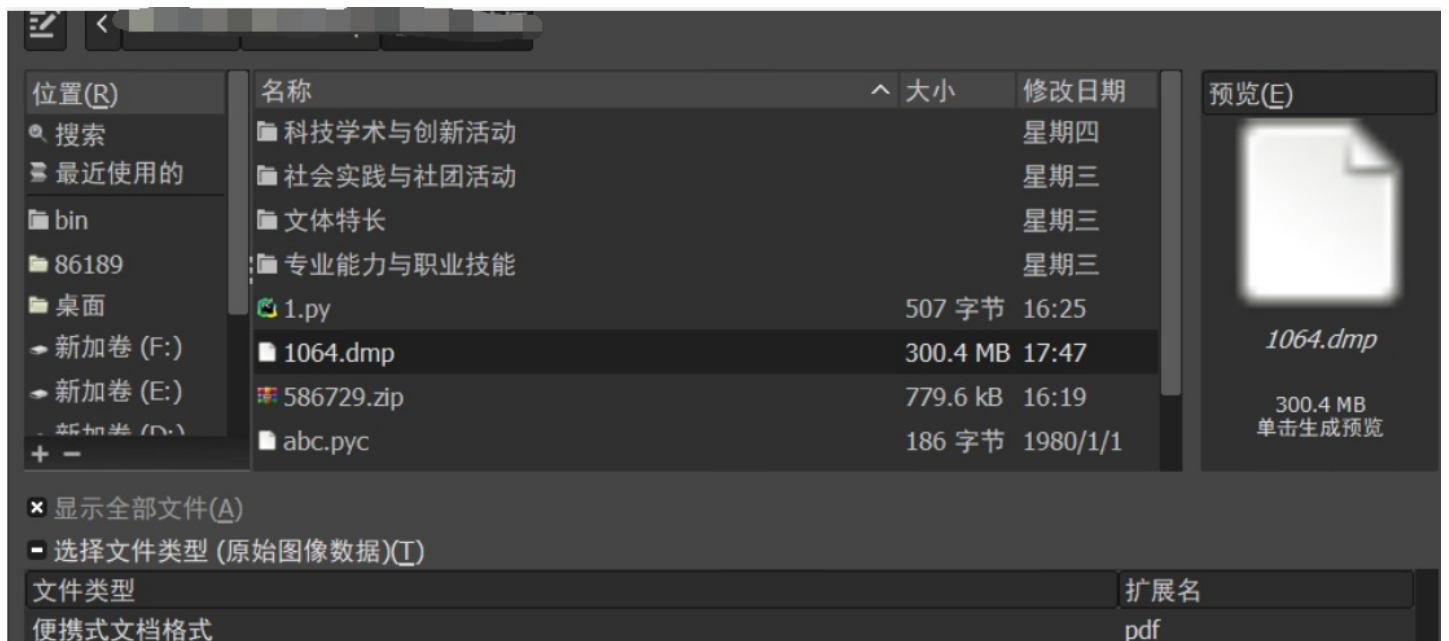
得到一个key，但是并不是压缩包的密码。

```
th1s_1s_th3_k3y
```

在看进程的时候有一个dumpit.exe,dump一下这个进程

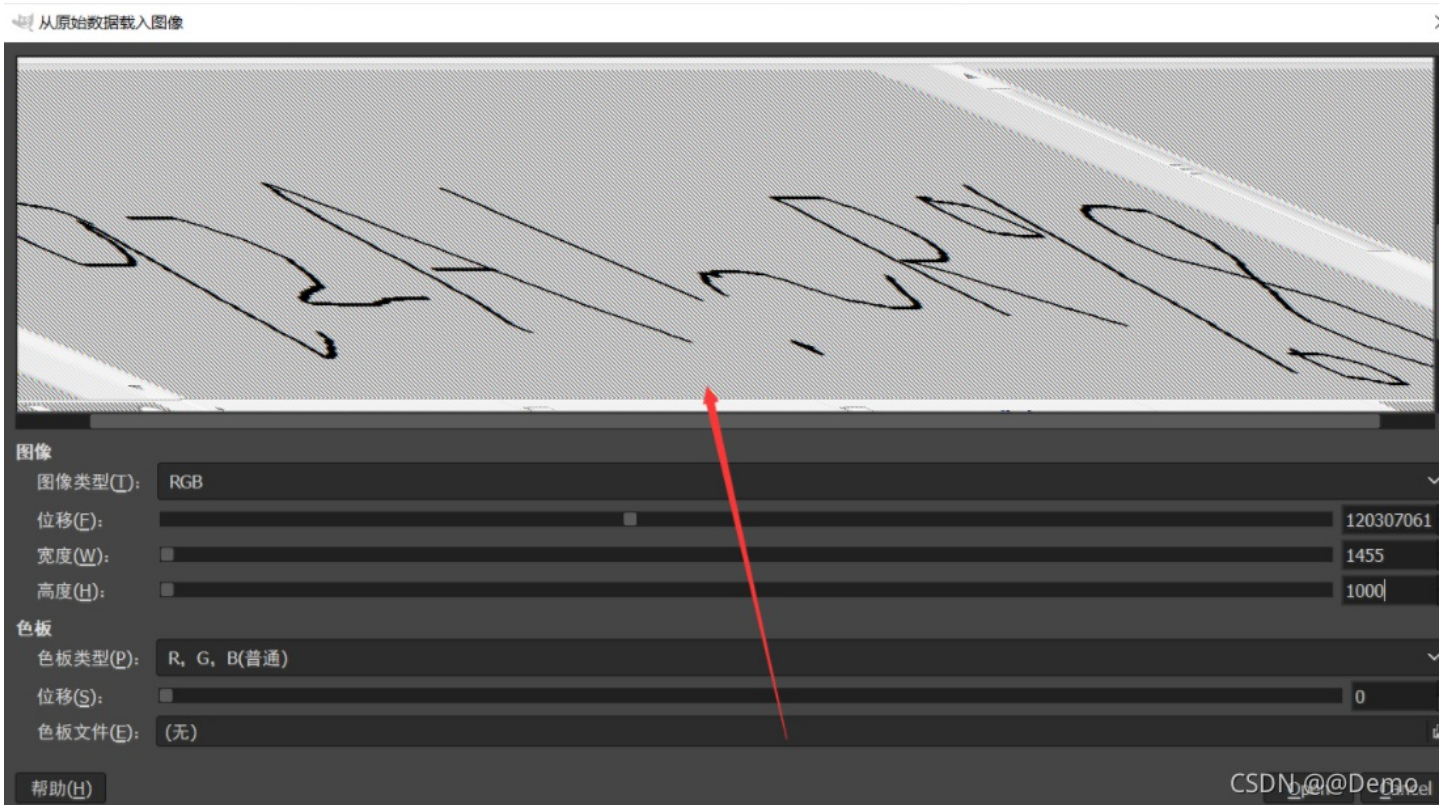
```
volatility -f mspaint.raw --profile=Win7SP1x64 memdump -p 1064 -D ./
```

```
root@kali:~/volatility_2.6_lin64_standalone# volatility -f mspaint.raw --profile=Win7SP1x64 memdump -p 1064 -D
Volatility Foundation Volatility Framework 2.6
*****
Writing DumpIt.exe [ 1064] to 1064.dmp
```





调整参数



得到压缩包的解压密码

q2A!~R%8

后面是队里的re手个人秀


```
key = 'th1s_1s_th3_k3y'
flag = ''
data = '12045014240343684450506E5E1E1C165D045E6B52113C5951006F091E4F4C0C54426A52466A165B0122'
data_list = []
for i in range(0,len(data),2):
    data_list.append('0x' + data[i:i+2])
print(data_list)
for i in range(0,len(data_list)):
    flag += chr(int(data_list[i],16)^ord(key[(i % 15)]))
print(flag)
```

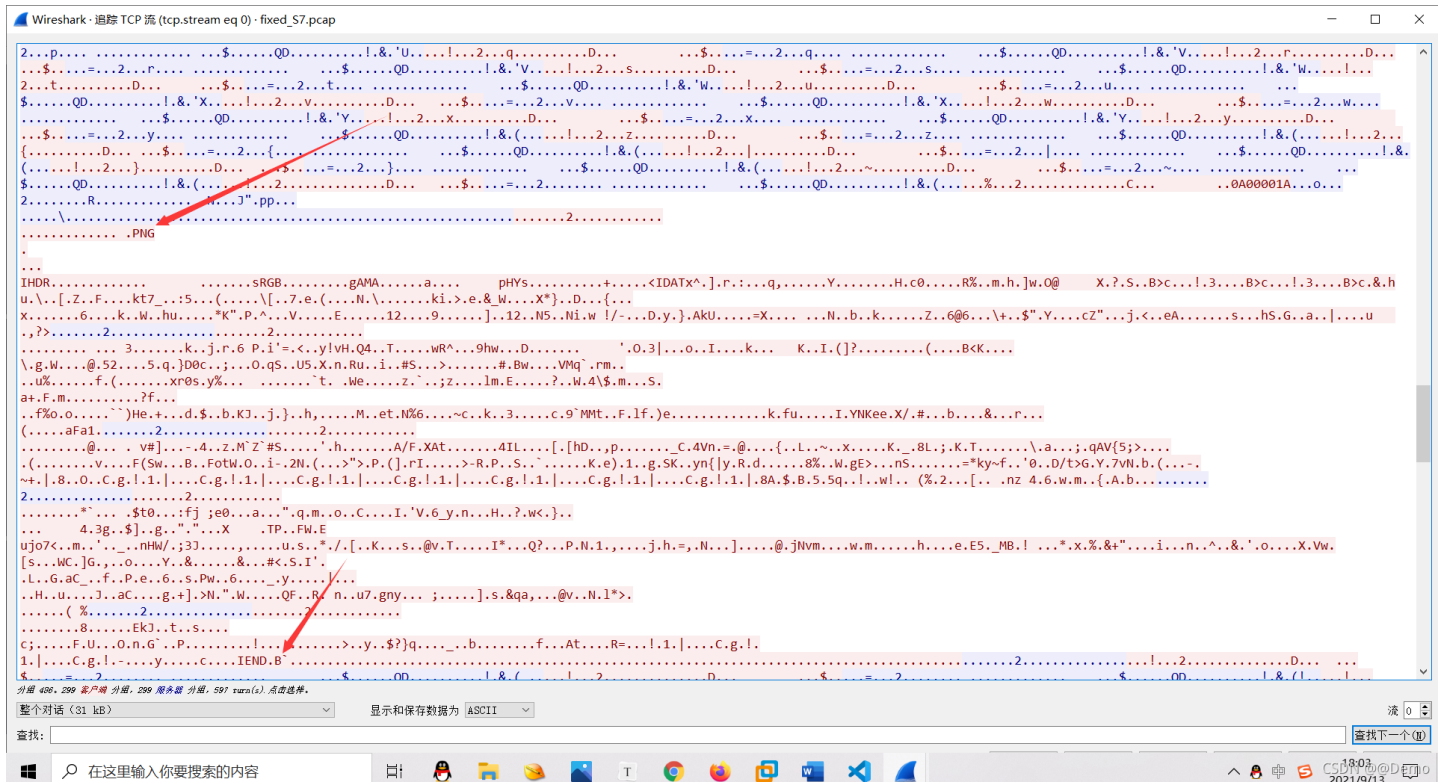
flag{20708c15-eb55-4cbc-930b-68de15c55b32}

藏在s7里的秘密

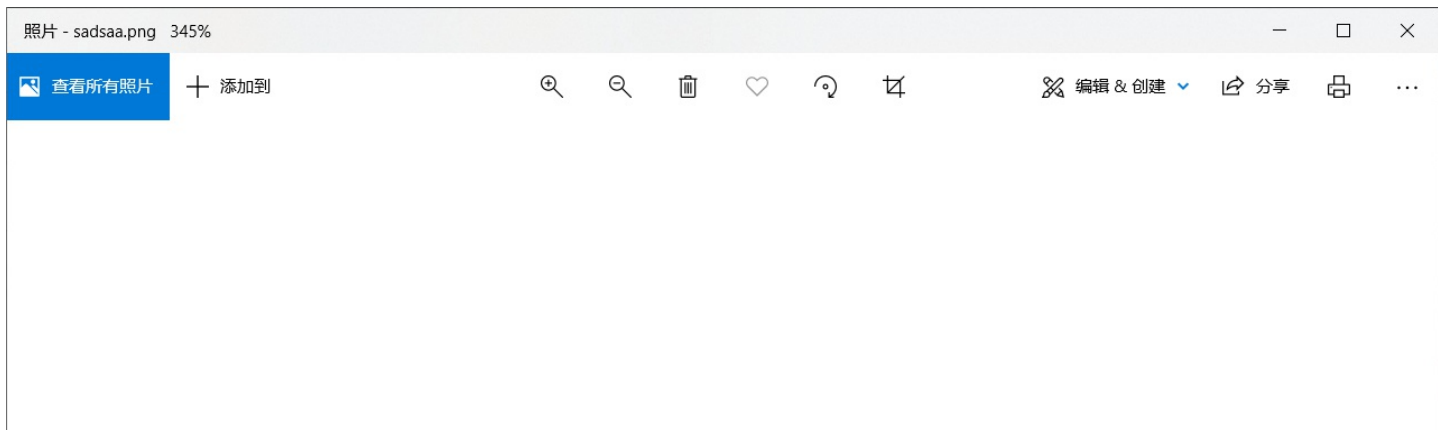
题目描述：某工厂的安全设备捕获了攻击者向PLC中写入恶意数据的数据包，你能分析出并找到其中隐藏的数据吗？

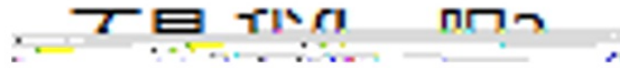
附件打开发现流量包损坏打不开，010打开发现文件头被修改了，直接pcapfix修复下就好。

打开后，追踪tcp流发现png图片



保存后发现只能看清楚一半的图





显然十六进制数据是不完整的，然后因为没咋做过流量包，到这思路也就停滞了。赛后看师傅的wp发现要看，这个流量包中出现最多的流s7comm，

先使用上面的过滤框，单独过滤这个协议

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.139.1	192.168.139.131	S7COMM	79	ROSCTR:[Job] Function:[Setup communication]
15	0.000000	192.168.139.131	192.168.139.1	S7COMM	81	ROSCTR:[Ack_Data] Function:[Setup communication]
17	0.000000	192.168.139.1	192.168.139.131	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
18	0.000000	192.168.139.131	192.168.139.1	S7COMM	207	ROSCTR:[Userdata] Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
19	0.000000	192.168.139.1	192.168.139.131	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0000
20	0.000000	192.168.139.131	192.168.139.1	S7COMM	435	ROSCTR:[Userdata] Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0000
21	0.000000	192.168.139.1	192.168.139.131	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0131 Index=0x0001
22	0.000000	192.168.139.131	192.168.139.1	S7COMM	135	ROSCTR:[Userdata] Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0131 Index=0x0001
31	0.000000	192.168.139.1	192.168.139.131	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
32	0.000000	192.168.139.131	192.168.139.1	S7COMM	115	ROSCTR:[Userdata] Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
34	0.000000	192.168.139.1	192.168.139.131	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
35	0.000000	192.168.139.131	192.168.139.1	S7COMM	115	ROSCTR:[Userdata] Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
38	0.000000	192.168.139.1	192.168.139.131	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000

在查看这个流的时候发现长度有不一样的地方，看第一个长度541的包，发现存在png的文件头，最后一个长度329的包存在png的文件尾

No.	Time	Source	Destination	Protocol	Length	Info
463	0.000000	192.168.139.1	192.168.139.131	S7COMM	87	ROSCTR:[Userdata] Function:[Request]
464	0.000000	192.168.139.131	192.168.139.1	S7COMM	115	ROSCTR:[Userdata] Function:[Response]
482	0.000000	192.168.139.1	192.168.139.131	S7COMM	91	ROSCTR:[Userdata] Function:[Request]
483	0.000000	192.168.139.131	192.168.139.1	S7COMM	165	ROSCTR:[Userdata] Function:[Response]
484	0.000000	192.168.139.1	192.168.139.131	S7COMM	541	ROSCTR:[Job] Function:[Write Va]
485	0.000000	192.168.139.131	192.168.139.1	S7COMM	76	ROSCTR:[Ack_Data] Function:[Write Va]
486	0.000000	192.168.139.1	192.168.139.131	S7COMM	541	ROSCTR:[Job] Function:[Write Va]
487	0.000000	192.168.139.131	192.168.139.1	S7COMM	76	ROSCTR:[Ack_Data] Function:[Write Va]
488	0.000000	192.168.139.1	192.168.139.131	S7COMM	541	ROSCTR:[Job] Function:[Write Va]
489	0.000000	192.168.139.131	192.168.139.1	S7COMM	76	ROSCTR:[Ack_Data] Function:[Write Va]

490 0.000000	192.168.139.1	192.168.139.131	S7COMM	541 ROSCTR:[Job]	Function:[Write Va
491 0.000000	192.168.139.131	192.168.139.1	S7COMM	76 ROSCTR:[Ack_Data]	Function:[Write Va
492 0.000000	192.168.139.1	192.168.139.131	S7COMM	329 ROSCTR:[Job]	Function:[Write Va

> Frame 219: 87 bytes on wire (696 bits). 87 bytes captured (696 bits)

导出第一部分举例
选中data->显示分组字节

The screenshot shows the Wireshark interface for a packet capture named 'fixed_S7.pcap'. The packet list pane shows several S7COMM packets. The packet details pane is expanded to show the 'S7 Communication' section, which includes a 'Data' field. A context menu is open over the 'Data' field, with the option '显示分组字节...' (Show packet bytes) highlighted. Other options include '导出分组字节流(B)...' (Export packet bytes stream), 'Wiki 协议页面' (Wiki protocol page), '过滤器字段参考' (Filter field reference), '协议首选项' (Protocol preferences), '解码为(A)...' (Decode as...), 'Go to Linked Packet' (Go to linked packet), and '在新窗口中显示已链接的分组' (Show linked packet in new window).

单击左下角改为原始数据

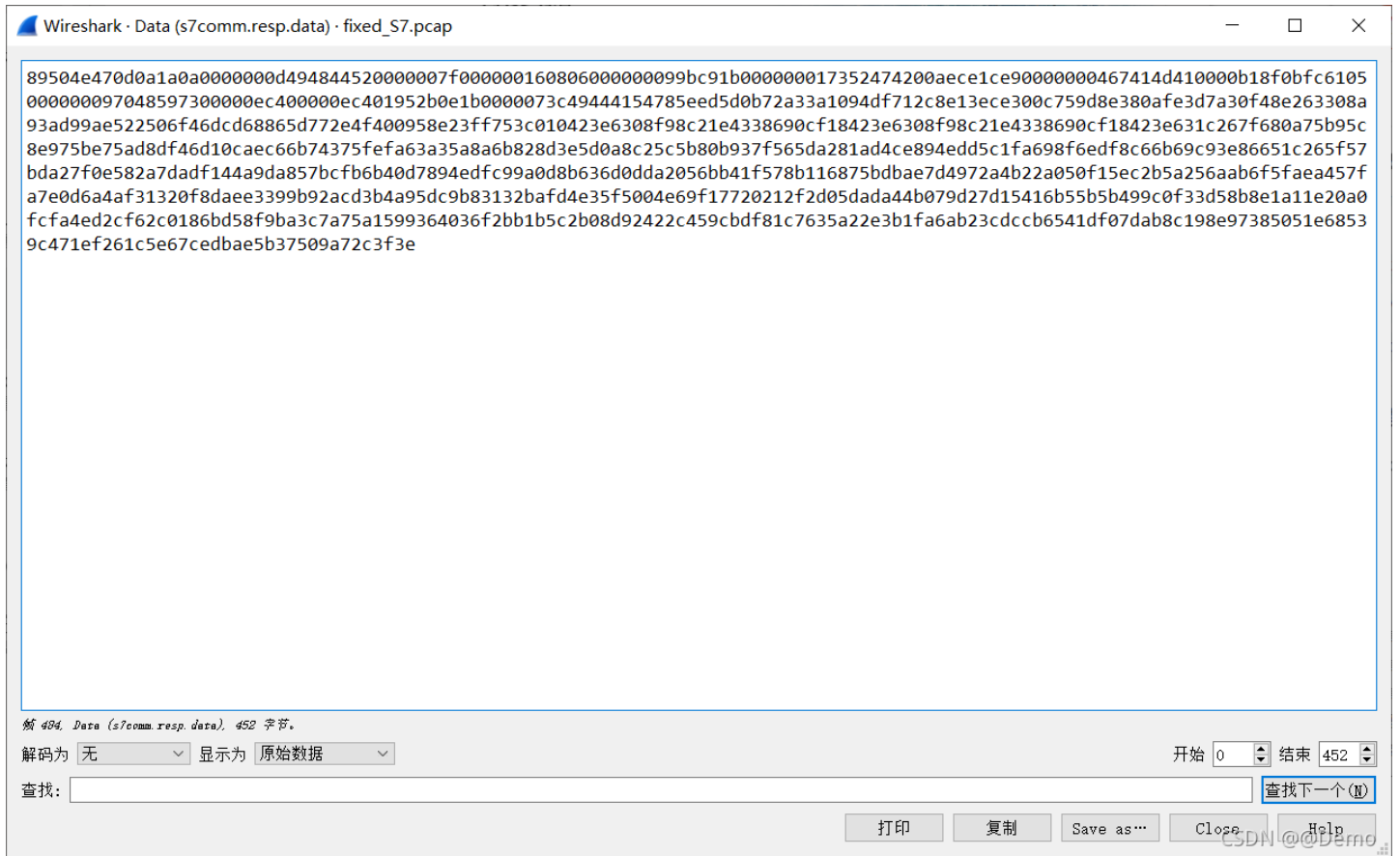
The screenshot shows the Wireshark interface with the 'Data' field expanded to show the raw bytes of the selected packet. The raw data is displayed in hexadecimal and ASCII. The ASCII view shows the following text:

```

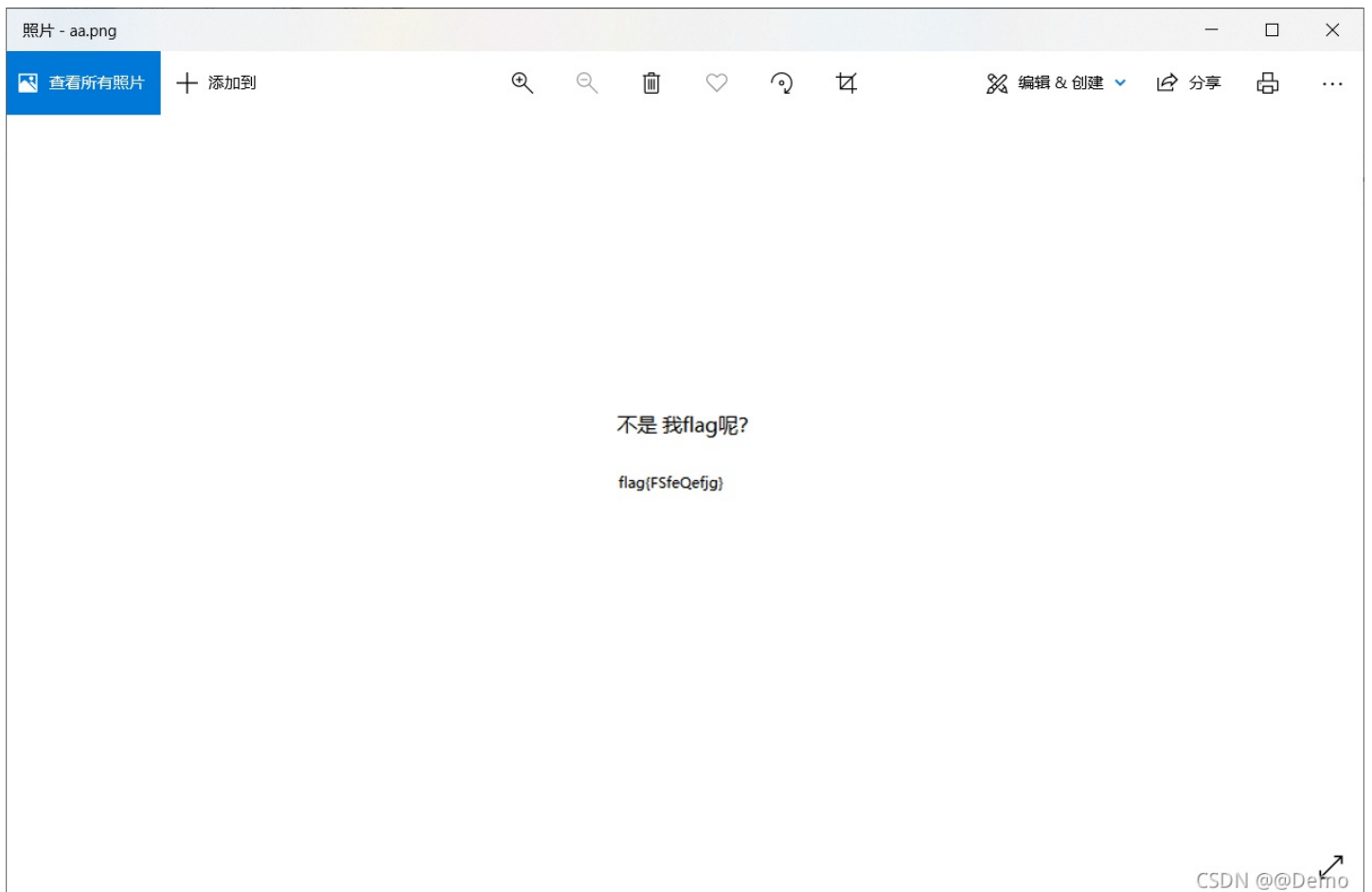
.PNG
.
IHDR . . . . .sRGB . . . . .gAMA . . . . .a. pHYs .. . . .+.. <IDATx^.]r.:...q,...
.Y.....H.c0....R%.m.h.]w.O@ X.?.S..B>c...!3...B>c...!3...B>c...h
u.\.[.Z.F. .kt7. :.5...(. . . . .\ [ .7.e.(. . . . .N.\. . . . .ki.)e.& W. . . . .X*}..D.\{... x.....
6...k..W..hu.....*K".P.^...V.....E.....12...9.....].12..N5. Ni.w !/-...D.y}.AKU.....=X.... .N..b.k.....Z..6@6... \
+..$.Y....cZ"...j.<..eA.....s...hS.G..a..|....u .,?>

```

The packet list pane at the bottom shows the selected packet: '帧 492, Data (s7comm.resp.data), 452 字节.' (Frame 492, Data (s7comm.resp.data), 452 bytes). The '解码为' (Decode as) dropdown is set to '无' (None) and the '显示为' (Show as) dropdown is set to 'ASCII'. The '开始' (Start) and '结束' (End) fields are set to 0 and 452 respectively. The '查找' (Search) field is empty, and the '查找下一个(N)' (Find next) button is visible.



选中010,ctrl+shift+v保存即可。按照顺序依次把数据加进去即可。
保存后发现crc的值报错，说明宽高被修改了，修改高度获得flag



```
flag{FSfeQefjg}
```

老练的黑客

题目描述：一黑客成功入侵某核电站且获得了操作员站控制权，该操作员站可控制离心机的转速，当离心机的转速持续高于5000时将导致设备损坏，为了保护设备，操作员站检测到转速超过5000时会自动限制转速。但该黑客非常老练，他在修改了转速后，还欺骗了操作员站，使得操作员站读取到错误的转速数据。你能找到黑客修改后的转速值和操作员站读取到的错误转速值吗？（flag格式：flag{修改后转速的+读取的错误转速} 值用16进制表示

通过查看发现转速的数据位于data

5000的十六进制为1388，所以只要找到data大于1388的包查看数据即可

使用过滤器

```
modbus.data>1388
```

realoldhacker.pcap

文件(F) 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(T) 帮助(H)

modbus.data>1338

No.	Time	Source	Destination	Protocol	Length	Info
1198	0.000000	192.168.3.130	192.168.3.164	Modbus...	66	Query: Trans: 29952; Unit: 1, Func: 6: Write Single
1199	0.000000	192.168.3.164	192.168.3.130	Modbus...	66	Response: Trans: 29952; Unit: 1, Func: 6: Write Single

> Frame 1199: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)

> Ethernet II, Src: VMware_6d:a1:06 (00:0c:29:6d:a1:06), Dst: VMware_cb:84:a0 (00:0c:29:cb:84:a0)

> Internet Protocol Version 4, Src: 192.168.3.164, Dst: 192.168.3.130

> Transmission Control Protocol, Src Port: 502, Dst Port: 12414, Seq: 2042, Ack: 1333, Len: 12

> Modbus/TCP

Modbus

.000 0110 = Function Code: Write Single Register (6)

[Request Frame: 1198]

[Time from request: 0.00000000 seconds]

Reference Number: 0

Data: 22b8

```

0000 00 0c 29 cb 84 a0 00 0c 29 6d a1 06 08 00 45 00  ..).....)m....E.
0010 00 34 28 ef 40 00 80 06 49 5e c0 a8 03 a4 c0 a8  -4(@...I^.....
0020 03 82 01 f6 30 7e b9 7d be 27 02 07 69 6b 50 18  ...0~}...'ikP.
0030 00 fb 77 fe 00 00 75 00 00 00 00 06 01 06 00 00  ..w...u.....
0040 22 b8                                     ".

```

CSDN @@Demo

所以第一部分的flag为22b8。

又说转速被修改了，往下翻倒数read的几个包，找到一个4500对应十六进制为1194,肯定是被修改的转速。

realoldhacker.pcap

文件(F) 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(T) 帮助(H)

应用显示过滤器 ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1284	0.000000	2a:6c:07:09:a4:cc	Broadcast	ARP	60	Who has 192.168.3.47? Tell 192.168.3.33
1285	0.000000	2a:6c:07:09:a4:cc	Broadcast	ARP	60	Who has 192.168.3.48? Tell 192.168.3.33
1286	0.000000	2a:6c:07:09:a4:cc	Broadcast	ARP	60	Who has 192.168.3.49? Tell 192.168.3.33
1287	0.000000	2a:6c:07:09:a4:cc	Broadcast	ARP	60	Who has 192.168.3.50? Tell 192.168.3.33
1288	0.000000	192.168.3.164	192.168.3.255	NBNS	92	Name query NB 1<00>
1289	0.000000	192.168.3.130	192.168.3.164	Modbus...	66	Query: Trans: 31744; Unit: 1, Func: 3: Read Holding Registers
1290	0.000000	192.168.3.164	192.168.3.130	Modbus...	73	Response: Trans: 31744; Unit: 1, Func: 3: Read Holding Registers
1291	0.000000	VMware_cb:84:a0	Broadcast	ARP	42	Who has 192.168.3.120? Tell 192.168.3.130
1292	0.000000	fe80::c15:c5e9:e6a2...	ff02::1:3	LLMNR	82	Standard query 0x7db1 A 66
1293	0.000000	192.168.3.164	224.0.0.252	LLMNR	62	Standard query 0x7db1 A 66
1294	0.000000	fe80::c15:c5e9:e6a2...	ff02::1:3	LLMNR	82	Standard query 0x7db1 A 66
1295	0.000000	192.168.3.164	224.0.0.252	LLMNR	62	Standard query 0x7db1 A 66
1296	0.000000	192.168.3.130	192.168.3.164	TCP	54	12414 → 502 [ACK] Seq=1417 Ack=2187 Win=391 Len=0

> Transmission Control Protocol, Src Port: 502, Dst Port: 12414, Seq: 2168, Ack: 1417, Len: 19

Modbus/TCP

Modbus

.000 0111 = Function Code: Read Holding Registers (3)

[Request Frame: 1289]

[Time from request: 0.00000000 seconds]

Byte Count: 10

> Register 0 (UINT16): 4500

> Register 1 (UINT16): 151

> Register 2 (UINT16): 101

> Register 3 (UINT16): 219

> Register 4 (UINT16): 232

```

0000 00 0c 29 cb 84 a0 00 0c 29 6d a1 06 08 00 45 00  ..).....)m....E.
0010 00 3b 28 f9 40 00 80 06 49 4d c0 a8 03 a4 c0 a8  ;(@...IM.....
0020 03 82 01 f6 30 7e b9 7d be a5 02 07 69 bf 50 18  ...0~}...'iP.
0030 01 00 35 c0 00 00 7c 00 00 00 0d 01 03 0a 11  ..5...|.....|
0040 34 00 97 00 65 00 db 00 e8                                     |...e...

```

Text item (text), 2 byte(s)

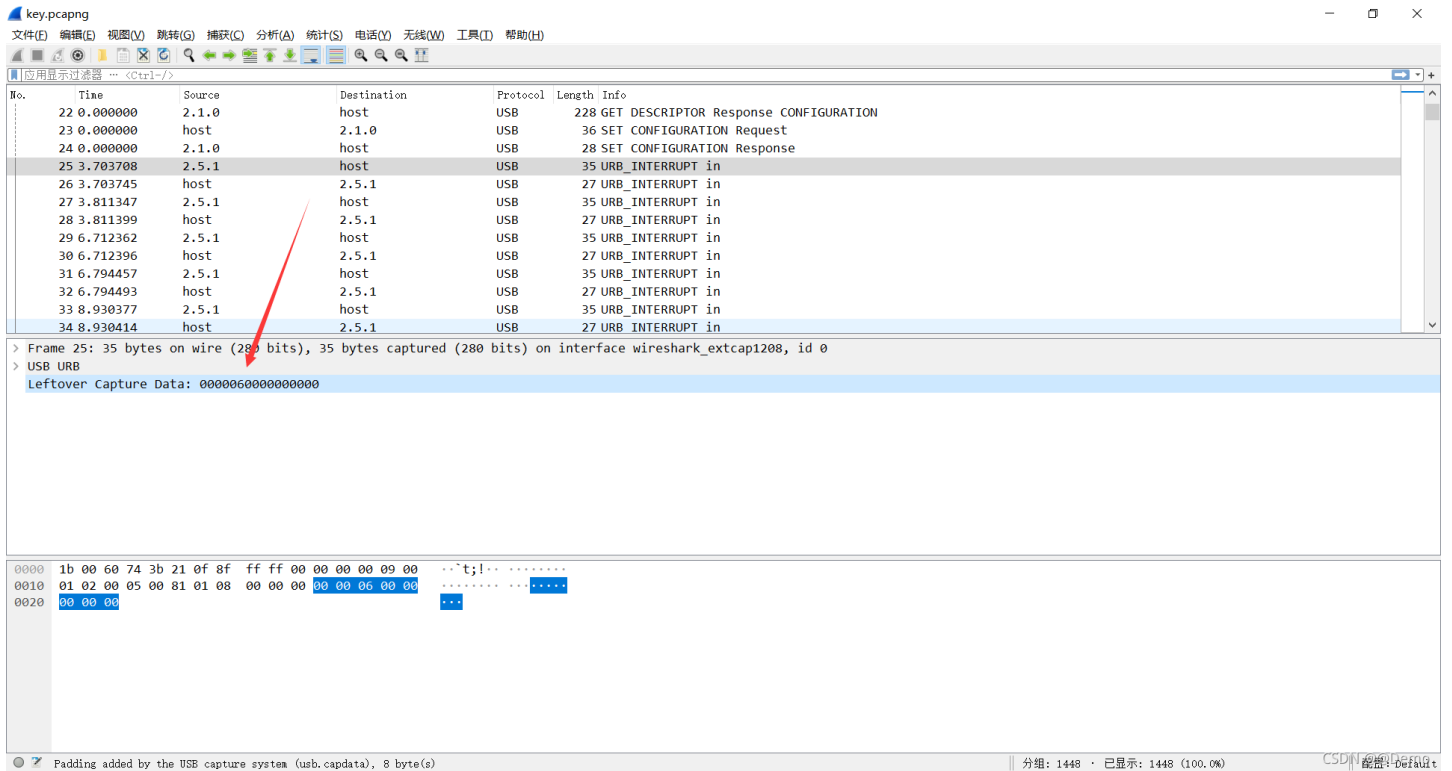
分組: 1410 · 已显示: 1410 (100.0%)

CSDN @@Demo


```
flag{22b81194}
```

baby-usb

usb流量找到有数据的包，发现是八字节的，可能是键盘流量。



key.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
22	0.000000	2.1.0	host	USB	228	GET_DESCRIPTOR Response CONFIGURATION
23	0.000000	host	2.1.0	USB	36	SET_CONFIGURATION Request
24	0.000000	2.1.0	host	USB	28	SET_CONFIGURATION Response
25	3.703708	2.5.1	host	USB	35	URB_INTERRUPT in
26	3.703745	host	2.5.1	USB	27	URB_INTERRUPT in
27	3.811347	2.5.1	host	USB	35	URB_INTERRUPT in
28	3.811399	host	2.5.1	USB	27	URB_INTERRUPT in
29	6.712362	2.5.1	host	USB	35	URB_INTERRUPT in
30	6.712396	host	2.5.1	USB	27	URB_INTERRUPT in
31	6.794457	2.5.1	host	USB	35	URB_INTERRUPT in
32	6.794493	host	2.5.1	USB	27	URB_INTERRUPT in
33	8.930377	2.5.1	host	USB	35	URB_INTERRUPT in
34	8.930414	host	2.5.1	USB	27	URB_INTERRUPT in

> Frame 25: 35 bytes on wire (280 bits), 35 bytes captured (280 bits) on interface wireshark_extcap1208, id 0
> USB URB
Leftover Capture Data: 0000600000000000

```
0000 1b 00 60 74 3b 21 0f 8f ff ff 00 00 00 09 00 ...t;! .....
0010 01 02 00 05 00 81 01 08 00 00 00 00 00 00 00 .....
0020 00 00 00
```

Padding added by the USB capture system (usb.capdata), 8 byte(s) | 分组: 1448 · 已显示: 1448 (100.0%)

那就找键盘流量的脚本梭一下。

用脚本直接跑没啥输出，所以先用tshark提取了一下

```
tshark -r key.pcapng -T fields -e usb.capdata | sed '/^\s*$/d' > fileaaa.txt
```

然后修改了下脚本有输出了

```
import os, sys

normalKeys = {
    "04": "a", "05": "b", "06": "c", "07": "d", "08": "e",
    "09": "f", "0a": "g", "0b": "h", "0c": "i", "0d": "j",
    "0e": "k", "0f": "l", "10": "m", "11": "n", "12": "o",
    "13": "p", "14": "q", "15": "r", "16": "s", "17": "t",
    "18": "u", "19": "v", "1a": "w", "1b": "x", "1c": "y",
    "1d": "z", "1e": "1", "1f": "2", "20": "3", "21": "4",
    "22": "5", "23": "6", "24": "7", "25": "8", "26": "9",
    "27": "0", "28": "<RET>", "29": "<ESC>", "2a": "<DEL>", "2b": "\t",
    "2c": "<SPACE>", "2d": "-", "2e": "=", "2f": "[", "30": "]", "31": "\\",
    "32": "<NON>", "33": ";", "34": "'", "35": "<GA>", "36": ",", "37": ".",
    "38": "/", "39": "<CAP>", "3a": "<F1>", "3b": "<F2>", "3c": "<F3>", "3d": "<F4>",
    "3e": "<F5>", "3f": "<F6>", "40": "<F7>", "41": "<F8>", "42": "<F9>", "43": "<F10>",
    "44": "<F11>", "45": "<F12>"
}

shiftKeys = {
    "04": "A", "05": "B", "06": "C", "07": "D", "08": "E",
    "09": "F", "0a": "G", "0b": "H", "0c": "I", "0d": "J",
    "0e": "K", "0f": "L", "10": "M", "11": "N", "12": "O",
    "13": "P", "14": "Q", "15": "R", "16": "S", "17": "T",
    "18": "U", "19": "V", "1a": "W", "1b": "X", "1c": "Y",
    "1d": "Z", "1e": "!", "1f": "@", "20": "#", "21": "$",
```

```

"22": "%", "23": "^", "24": "&", "25": "*", "26": "(", "27": ")",
"28": "<RET>", "29": "<ESC>", "2a": "<DEL>", "2b": "\\t", "2c": "<SPACE>",
"2d": "_", "2e": "+", "2f": "{", "30": "}", "31": "|", "32": "<NON>", "33": "\\\"",
"34": ":", "35": "<GA>", "36": "<", "37": ">", "38": "?", "39": "<CAP>", "3a": "<F1>",
"3b": "<F2>", "3c": "<F3>", "3d": "<F4>", "3e": "<F5>", "3f": "<F6>", "40": "<F7>",
"41": "<F8>", "42": "<F9>", "43": "<F10>", "44": "<F11>", "45": "<F12>"}

#pcapFilePath = sys.argv[1]
#os.system("tshark -r "+pcapFilePath+ " -T fields -e usb.capdata | sed '/^\s*$/d' > out.txt")

output = []
keys = open('out.txt')
for line in keys:
    line = ''.join(line[i:i+2]+':' for i in range(0,len(line)-1,2)).strip(':')
    try:
        if line[0]!='0' or (line[1]!='0' and line[1]!='2') or line[3]!='0' or line[4]!='0' or line[9]!='0' or line[10]!='0' or line[12]!='0' or line[13]!='0' or line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0' or line[21]!='0' or line[22]!='0' or line[6:8]=="00":
            continue
        if line[6:8] in normalKeys.keys():
            output += [[normalKeys[line[6:8]], [shiftKeys[line[6:8]]][line[1]=='2']]
        else:
            output += ['[unknown]']
    except:
        pass

keys.close()

flag=0
#print("".join(output))
for i in range(len(output)):
    try:
        a=output.index('<DEL>')
        del output[a]
        del output[a-1]
    except:
        pass

for i in range(len(output)):
    try:
        if output[i]=="<CAP>":
            flag+=1
            output.pop(i)
            if flag==2:
                flag=0
        if flag!=0:
            output[i]=output[i].upper()
    except:
        pass

print ('output : ' + "".join(output))
os.system("rm -rf out.txt")

```

output :congratulationsonfindingmebutiwillnottellyouwherethepasswordofworddocumentisgoandfinditagain

到这其实就没什么思路了，尝试了各种东西，赛后看发现方向错了。

借用盖乐希师傅的脚本

```
normalKeys = {
```

```

"04": "a", "05": "b", "06": "c", "07": "d", "08": "e",
"09": "f", "0a": "g", "0b": "h", "0c": "i", "0d": "j",
"0e": "k", "0f": "l", "10": "m", "11": "n", "12": "o",
"13": "p", "14": "q", "15": "r", "16": "s", "17": "t",
"18": "u", "19": "v", "1a": "w", "1b": "x", "1c": "y",
"1d": "z", "1e": "1", "1f": "2", "20": "3", "21": "4",
"22": "5", "23": "6", "24": "7", "25": "8", "26": "9",
"27": "0", "28": "<RET>", "29": "<ESC>", "2a": "<DEL>", "2b": "\t",
"2c": "<SPACE>", "2d": "-", "2e": "=", "2f": "[", "30": "]", "31": "\\ ",
"32": "<NON>", "33": ";", "34": "'", "35": "<GA>", "36": ",", "37": ".",
"38": "/", "39": "<CAP>", "3a": "<F1>", "3b": "<F2>", "3c": "<F3>", "3d": "<F4>",
"3e": "<F5>", "3f": "<F6>", "40": "<F7>", "41": "<F8>", "42": "<F9>", "43": "<F10>",
"44": "<F11>", "45": "<F12>"}
shiftKeys = {
"04": "A", "05": "B", "06": "C", "07": "D", "08": "E",
"09": "F", "0a": "G", "0b": "H", "0c": "I", "0d": "J",
"0e": "K", "0f": "L", "10": "M", "11": "N", "12": "O",
"13": "P", "14": "Q", "15": "R", "16": "S", "17": "T",
"18": "U", "19": "V", "1a": "W", "1b": "X", "1c": "Y",
"1d": "Z", "1e": "!", "1f": "@", "20": "#", "21": "$",
"22": "%", "23": "^", "24": "&", "25": "*", "26": "(", "27": ")",
"28": "<RET>", "29": "<ESC>", "2a": "<DEL>", "2b": "\t", "2c": "<SPACE>",
"2d": "_", "2e": "+", "2f": "{", "30": "}", "31": "|", "32": "<NON>", "33": "\ ",
"34": ":", "35": "<GA>", "36": "<", "37": ">", "38": "?", "39": "<CAP>", "3a": "<F1>",
"3b": "<F2>", "3c": "<F3>", "3d": "<F4>", "3e": "<F5>", "3f": "<F6>", "40": "<F7>",
"41": "<F8>", "42": "<F9>", "43": "<F10>", "44": "<F11>", "45": "<F12>"}

output = []
keys = open('filebbb.txt')
for line in keys:
    try:
        if line[0]!='0' or (line[1]!='0' and line[1]!='2') or line[3]!='0' or line[4]!='0' or line[9]!='0' or line[10]!='0' or line[12]!='0' or line[13]!='0' or line[15]!='0' or line[16]!='0' or line[18]!='0' or line[19]!='0' or line[21]!='0' or line[22]!='0' or line[6:8]=="00":
            continue
        if line[6:8] in normalKeys.keys():
            output += [[normalKeys[line[6:8]], [shiftKeys[line[6:8]]][line[1]=='2']]
        else:
            output += ['[unknown]']
    except:
        pass

keys.close()

flag=0
print("".join(output))
for i in range(len(output)):
    try:
        a=output.index('<DEL>')
        del output[a]
        del output[a-1]
    except:
        pass

for i in range(len(output)):
    try:
        if output[i]=="<CAP>":
            flag+=1
            output.pop(i)
        if flag==2:
            flag=0

```



```
    flag=0
    if flag!=0:
        output[i]=output[i].upper()
    except:
        pass

print ('output :' + "".join(output))
```

```
ct<DEL>onh<DEL>gratue<DEL>latoi<DEL><DEL>nsonfiny<DEL>dingmebutiwii<DEL>llns<DEL>ottellyouwheretqa<DEL><DEL>he
pz<DEL>asswords<DEL><DEL>ox<DEL>fwe<DEL>od<DEL>rddoc<DEL>cumentisgoarf<DEL><DEL><DEL>ndfinditagain
output :congratulationsonfindingmebutiwillnottellyouwherethepasswordofworddocumentisgoandfinditagain
```

每个前面的字符就是key qazwsxedcfrv

```
flag{685b42b0-da3d-47f4-a76c-0f3d07ea962a}
```