

# ctf流量分析练习二

原创

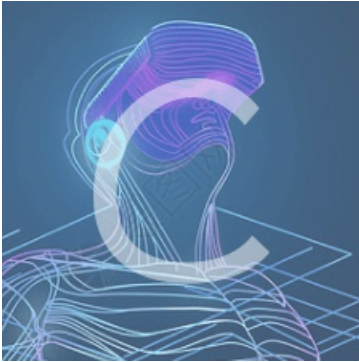
gclome 于 2020-09-06 17:53:01 发布 1535 收藏 15

分类专栏: #CTF

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\\_44108455/article/details/108179086](https://blog.csdn.net/qq_44108455/article/details/108179086)

版权



[CTF 专栏收录该内容](#)

20 篇文章 0 订阅

订阅专栏

上次的流量分析做的我一个脑袋两个大! 但是不能放弃啊, 再找一些题来练练手

## 0x01 经典题型

CTF题型主要分为流量包修复、WEB流量包分析、USB流量包分析和其他流量包分析。

### 01 流量包修复

比赛过程中有可能会通过wireshark打开题目给的流量包后提示包异常的情况, 如下图所示:

No.	Time	Source	Destination	Protocol	Length	Text item	Info
1	0.000000	172.16.0.29	91.189.89.198	NTP	90	✓	NTP Version 4, client
2	0.315107	91.189.89.198	172.16.0.29	NTP	90	✓	NTP Version 4, server
3	0.353186	127.0.0.1	127.0.0.1	DNS	72	✓	Standard query 0x348b A www.bing.com
4	0.353242	127.0.0.1	127.0.0.1	DNS	72	✓	Standard query 0x348b A www.bing.com
5	0.353613	172.16.0.29	172.16.0.1	DNS	72	✓	Standard query 0x1b9f A www.bing.com
6	0.385069	172.16.0.1	172.16.0.29	DNS	140	✓	Standard query response 0x1b9f A www.bing.com CNAME cn.a-0001.a
7	0.385263	127.0.0.1	127.0.0.1	DNS	140	✓	Standard query response 0x348b A www.bing.com CNAME cn.a-0001.a
8	0.385268	127.0.0.1	127.0.0.1	DNS	140	✓	Standard query response 0x348b A www.bing.com CNAME cn.a-0001.a

解题思路:

通过在线pcap包修复工具进行修复。

修复之后，再次打开

用tcp contains “flag”，找到了下面这句话：

YlqematYsQWMZ5Cn.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)

tcp contains "flag"

No.	Time	Source	Destination	Protocol	Length	Text item	Info
987	5.933360	172.16.0.29	202.89.233.103	HTTP	826	✓	GET /AS/Suggestions?pt=page.home&mkt=zh-cn&qry=where%20is%20my%20...
1159	18.234852	172.16.0.29	172.16.0.29	TCP	72	✓	999 → 2222 [SYN] Seq=0 Win=8192 Len=18
1160	18.234945	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 999 → 2222 [SYN] Seq=0 Win=8192 Len=18
1161	18.236163	172.16.0.29	172.16.0.29	TCP	72	✓	44247 → 2222 [SYN] Seq=0 Win=8192 Len=18
1162	18.236166	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 44247 → 2222 [SYN] Seq=0 Win=8192 Len=18

> Frame 1159: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)

> Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

> Internet Protocol Version 4, Src: 172.16.0.29, Dst: 172.16.0.29

> Transmission Control Protocol, Src Port: 999, Dst Port: 2222, Seq: 0, Len: 18

> Data (18 bytes)

Data: 77686572652069732074686520666c61673f

[Length: 18]

```

0000 ff ff ff ff ff ff 00 00 00 00 08 00 45 00 .....E.
0010 00 3a 6c 66 00 00 40 06 b5 fd ac 10 00 1d ac 10 :!f@.....
0020 00 1d 03 e7 08 ae 00 00 00 00 00 00 50 02 .....P.
0030 20 00 02 92 00 00 77 68 65 72 65 20 69 73 20 74 .....where is t
0040 68 65 20 66 6c 61 67 3f                          he flag?
  
```

在这个包里发现了这句提醒的话

于是乎，flag就在下面十几个tcp数据包里

No.	Time	Source	Destination	Protocol	Length	Text item	Info
1159	18.234852	172.16.0.29	172.16.0.29	TCP	72	✓	999 → 2222 [SYN] Seq=0 Win=8192 Len=18
1160	18.234945	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 999 → 2222 [SYN] Seq=0 Win=8192 Len=18
1161	18.236163	172.16.0.29	172.16.0.29	TCP	72	✓	44247 → 2222 [SYN] Seq=0 Win=8192 Len=18
1162	18.236166	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 44247 → 2222 [SYN] Seq=0 Win=8192 Len=18
1163	18.237200	172.16.0.29	172.16.0.29	TCP	72	✓	62457 → 2222 [SYN] Seq=0 Win=8192 Len=18
1164	18.237202	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 62457 → 2222 [SYN] Seq=0 Win=8192 Len=18
1165	18.238256	172.16.0.29	172.16.0.29	TCP	72	✓	29828 → 2222 [SYN] Seq=0 Win=8192 Len=18
1166	18.238259	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 29828 → 2222 [SYN] Seq=0 Win=8192 Len=18
1167	18.239363	172.16.0.29	172.16.0.29	TCP	72	✓	26374 → 2222 [SYN] Seq=0 Win=8192 Len=18
1168	18.239365	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 26374 → 2222 [SYN] Seq=0 Win=8192 Len=18
1169	18.240542	172.16.0.29	172.16.0.29	TCP	72	✓	46016 → 2222 [SYN] Seq=0 Win=8192 Len=18
1170	18.240545	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 46016 → 2222 [SYN] Seq=0 Win=8192 Len=18
1171	18.241728	172.16.0.29	172.16.0.29	TCP	72	✓	7989 → 2222 [SYN] Seq=0 Win=8192 Len=18
1172	18.241730	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 7989 → 2222 [SYN] Seq=0 Win=8192 Len=18
1173	18.243199	172.16.0.29	172.16.0.29	TCP	72	✓	43322 → 2222 [SYN] Seq=0 Win=8192 Len=18
1174	18.243202	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 43322 → 2222 [SYN] Seq=0 Win=8192 Len=18
1175	18.244501	172.16.0.29	172.16.0.29	TCP	72	✓	5661 → 2222 [SYN] Seq=0 Win=8192 Len=18
1176	18.244504	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 5661 → 2222 [SYN] Seq=0 Win=8192 Len=18
1177	18.245711	172.16.0.29	172.16.0.29	TCP	72	✓	1658 → 2222 [SYN] Seq=0 Win=8192 Len=18
1178	18.245714	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 1658 → 2222 [SYN] Seq=0 Win=8192 Len=18
1179	18.247424	172.16.0.29	172.16.0.29	TCP	72	✓	10975 → 2222 [SYN] Seq=0 Win=8192 Len=18
1180	18.247428	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 10975 → 2222 [SYN] Seq=0 Win=8192 Len=18
1181	18.248581	172.16.0.29	172.16.0.29	TCP	72	✓	32649 → 2222 [SYN] Seq=0 Win=8192 Len=18
1182	18.248584	172.16.0.29	172.16.0.29	TCP	72	✓	[TCP Out-Of-Order] 32649 → 2222 [SYN] Seq=0 Win=8192 Len=18
1183	18.250081	172.16.0.29	172.16.0.29	TCP	72	✓	43986 → 2222 [SYN] Seq=0 Win=8192 Len=18

注意到这十几个相连的数据包的ip头中的id字段

YlqematYsQWMZ5Cn.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)

tcp contains "flag"

Wireshark - 分组 1162 - YlqematYsQWMZ5Cn.pcap

> Frame 1162: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)

> Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

> Internet Protocol Version 4, Src: 172.16.0.29, Dst: 172.16.0.29

0100 .... = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 58

Identification: 0x6761 (26465)

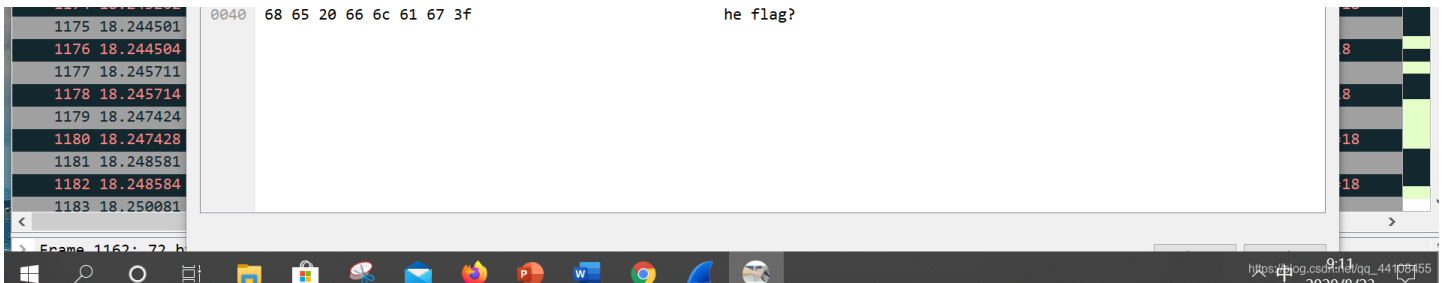
Flags: 0x0000

Time to live: 64

Protocol: TCP (6)

```

0000 ff ff ff ff ff ff 00 00 00 00 08 00 45 00 .....E.
0010 00 3a 67 61 00 00 40 06 bb 02 ac 10 00 1d ac 10 :ga@.....
0020 00 1d ac d7 08 ae 00 00 00 00 00 00 50 02 .....P.
0030 20 00 59 a1 00 00 77 68 65 72 65 20 69 73 20 74 .....Y...where is t
  
```



看起来一些有意义的字符

将全部这种数据包的id字段提取出来之后按lsb的方式组合起来就是flag;

这里呢，我去百度了一下lsb是什么，然后在参照原题目作者的意思，就是将每个包里的id的最后一位数取出来然后拼接在一起就是flag!

## 最低有效位

LSB，英文 least significant bit，中文译最低有效位。

对于一个给定的数据串（整数），如二进制的1001或者十进制351，其最低有效位就是拥有最小单位数值的那一位。比如二进制1001的最右一位，拥有数值1，在该整数中代表最低位，该位的值可以决定整数是奇数（为1）还是偶数（为0）。十进制数同理。

一般lsb就是一个整数的最右一位，所以似乎该概念有些多余。但是凡事都有例外，某些数据传输或是处理器恰恰相反，最左一位是lsb，所以在计算领域就定义了这个最低有效位以明确说明数据格式。

(LSB: Least Significant Byte)最低有效字节

其意义和lsb类似，只是扩展到整个字节，以字节为最小单位来说明数据的顺序。

[https://blog.csdn.net/qq\\_44108455](https://blog.csdn.net/qq_44108455)

pacp文件地址：<https://static2.ichunqiu.com/icq/resources/fileupload/CTF/BSRC/2017/BSRC3-1/findtheflag.cap>

## 02 WEB流量包分析

WEB数据包分析的题目主要出现WEB攻击行为的分析上，典型的WEB攻击行为有：WEB扫描、后台目录爆破、后台账号爆破、WEBSHELL上传、SQL注入等等。

### 001 WEB扫描分析

这部分就是用了安恒八月月赛流量分析的这道题，这道题我看了一个writeup，真的写的很好：<https://www.cnblogs.com/sn1per/p/12553064.html>，我也转载到了自己的博客中。

题型：

通过给出的流量包获取攻击者使用的WEB扫描工具。

解题思路：

常见的WEB扫描器有Awww, Netsparker, Appscan, Webinspect, RsaS (绿盟极光), Nessus, WebReaver, Sqlmap等。要识别攻击者使用的是哪一种扫描器，可通过wireshark筛选扫描器特征来得知。

相关命令：`http contains "扫描器特征值"`。

常见的扫描器特征参考：<https://www.freebuf.com/column/156291.html>

题目：

安恒八月月赛流量分析：黑客使用的是哪种扫描器？

pacp文件地址：

链接：<https://pan.baidu.com/s/1bGEIPeXDCbhybmWOyGr8Og> 提取码：q6ro

### 002 后台账号爆破

题型：

已知攻击者通过暴力破解的手段获取了网站的后台登陆账号，请通过给出的流量包获取正确的账号信息。

解题思路：

WEB账号登陆页面通常采用post方法请求，要获取流量包中记录的账号信息可通过wireshark筛选出POST请求和账号中的关键字如'admin'。

相关命令：`http.request.method=="POST" && http.contains=="关键字"`。

练练手：

安恒八月月赛流量分析：黑客使用了什么账号密码登录了web后台？

### 003 WEBSHELL上传

题型：

已知攻击者上传了恶意webshell文件，请通过给出的流量包还原出攻击者上传的webshell内容。

解题思路：

Webshell文件上传常采用post方法请求，文件内容常见关键字eval, system, assert要。获取流量包中记录的webshell可通过wireshark筛选出POST请求和关键字。

相关命令：`http.request.method=="POST" && http.contains=="关键字"`

练练手：

安恒八月月赛流量分析：黑客上传的webshell文件名是？内容是什么？

## 03 USB流量包分析

USB流量指的是USB设备接口的流量，攻击者能够通过监听usb接口流量获取键盘敲击键、鼠标移动与点击、存储设备的铭文传输通信、USB无线网卡网络传输内容等等。在CTF中，USB流量分析主要以键盘和鼠标流量为主。

### 001键盘流量

USB协议数据部分在Leftover Capture Data域中，数据长度为八个字节。其中键盘击键信息集中在第三个字节中。

题型：

Flag藏于usb流量中，通过USB协议数据中的键盘键码转换成键位。

解题思路：

1.使用kali linux中的tshark 命令把cap data提取出来：`tshark -r usb.pcap -T fields -e usb.capdata > usbdata.txt`，并去除空行。

2. 根据《USB键盘协议中键码》中的HID Usage ID将数据还原成键位。

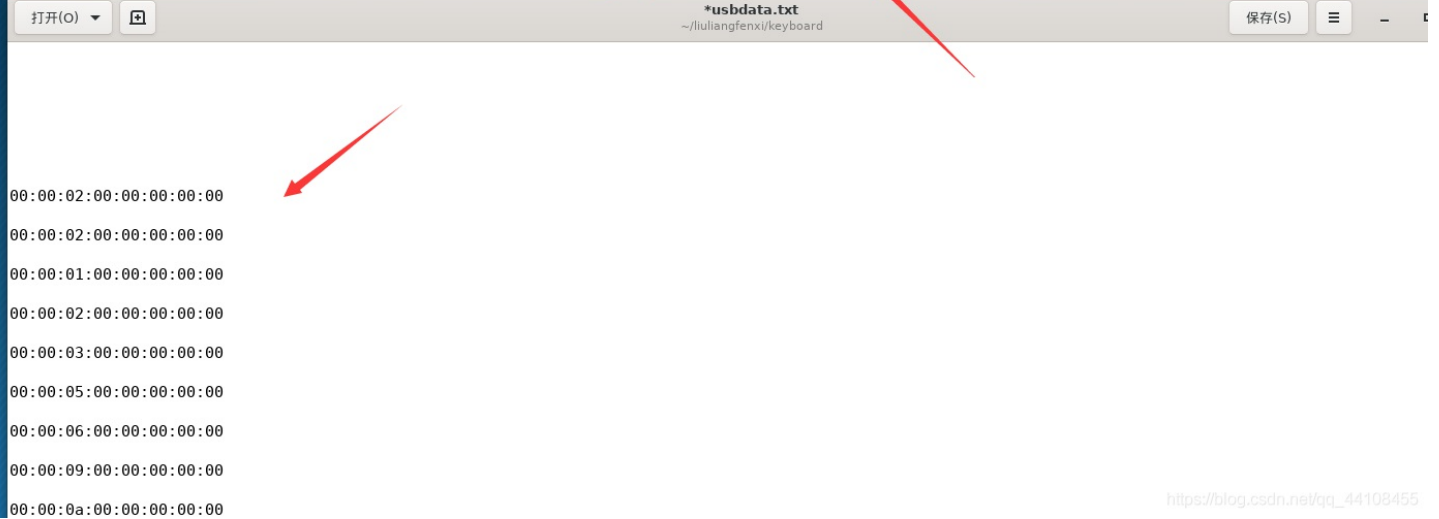
练练手：

安全评测人员在对某银行卡密码输入系统进行渗透测试，截获了一段通过USB键盘输入6位数字密码的流量，其中也包含了一些其他无关的USB设备的流量，你能从中恢复出6位数字密码吗？最终提交的flag格式为flag。

使用tshark 命令把cap data提取出来：

```
tshark -r usb.pcap -T fields -e usb.capdata > usbdata.txt
```

```
root@kali:~/liuliangfenxi/keyboard# tshark -r usb.pcap -T fields -e usb.capdata > usbdata.txt
Running as user "root" and group "root". This could be dangerous.
root@kali:~/liuliangfenxi/keyboard#
```



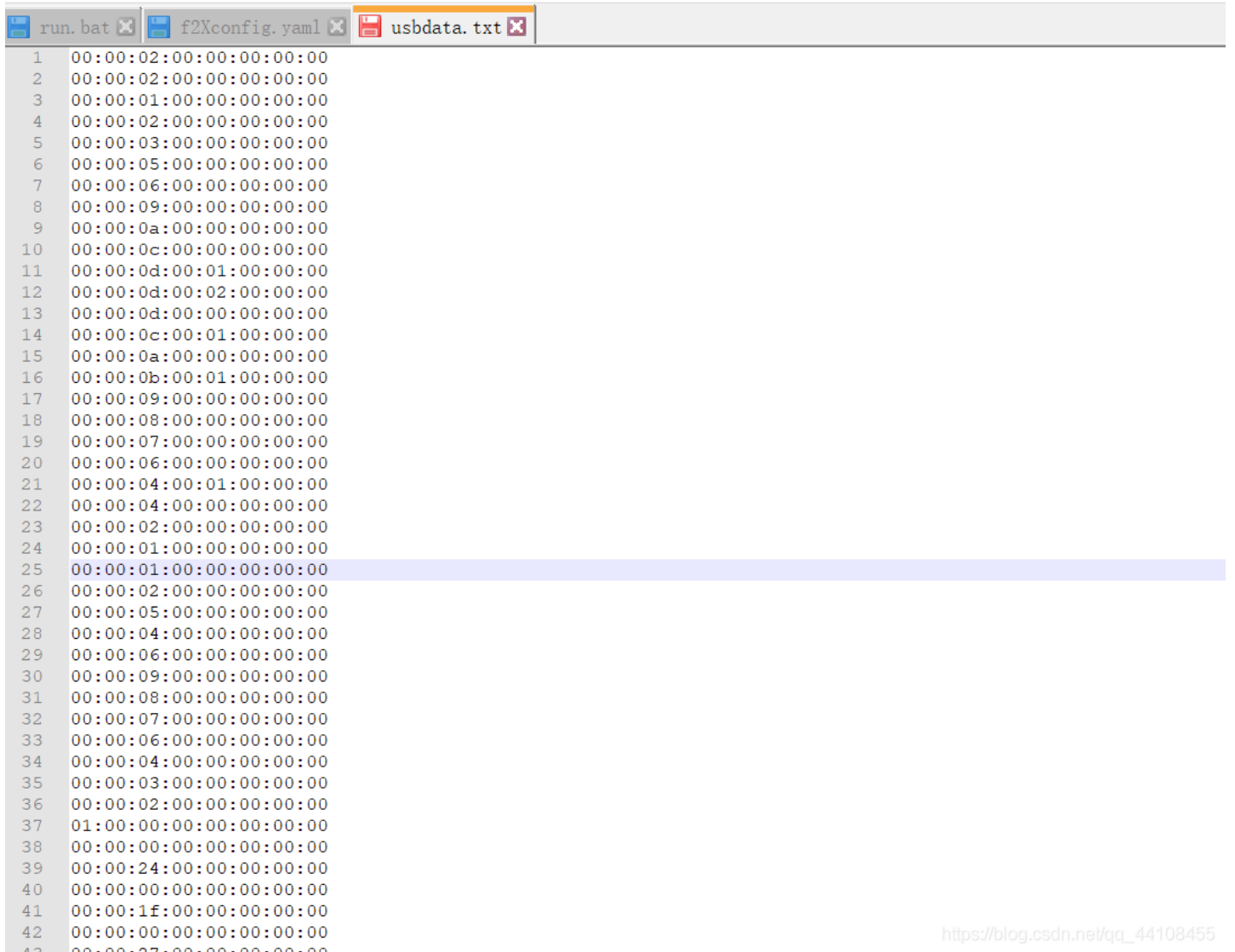
```
*usbdata.txt
~/liuliangfenxi/keyboard

00:00:02:00:00:00:00
00:00:02:00:00:00:00
00:00:01:00:00:00:00
00:00:02:00:00:00:00
00:00:03:00:00:00:00
00:00:05:00:00:00:00
00:00:06:00:00:00:00
00:00:09:00:00:00:00
00:00:0a:00:00:00:00
```

[https://blog.csdn.net/qq\\_44108455](https://blog.csdn.net/qq_44108455)

去除空行:

方法: 用notepad++打开, 编辑->行操作->移除空行



```
run.bat x f2Xconfig.yaml x usbdata.txt x
1 00:00:02:00:00:00:00
2 00:00:02:00:00:00:00
3 00:00:01:00:00:00:00
4 00:00:02:00:00:00:00
5 00:00:03:00:00:00:00
6 00:00:05:00:00:00:00
7 00:00:06:00:00:00:00
8 00:00:09:00:00:00:00
9 00:00:0a:00:00:00:00
10 00:00:0c:00:00:00:00
11 00:00:0d:00:01:00:00:00
12 00:00:0d:00:02:00:00:00
13 00:00:0d:00:00:00:00:00
14 00:00:0c:00:01:00:00:00
15 00:00:0a:00:00:00:00:00
16 00:00:0b:00:01:00:00:00
17 00:00:09:00:00:00:00:00
18 00:00:08:00:00:00:00:00
19 00:00:07:00:00:00:00:00
20 00:00:06:00:00:00:00:00
21 00:00:04:00:01:00:00:00
22 00:00:04:00:00:00:00:00
23 00:00:02:00:00:00:00:00
24 00:00:01:00:00:00:00:00
25 00:00:01:00:00:00:00:00
26 00:00:02:00:00:00:00:00
27 00:00:05:00:00:00:00:00
28 00:00:04:00:00:00:00:00
29 00:00:06:00:00:00:00:00
30 00:00:09:00:00:00:00:00
31 00:00:08:00:00:00:00:00
32 00:00:07:00:00:00:00:00
33 00:00:06:00:00:00:00:00
34 00:00:04:00:00:00:00:00
35 00:00:03:00:00:00:00:00
36 00:00:02:00:00:00:00:00
37 01:00:00:00:00:00:00:00
38 00:00:00:00:00:00:00:00
39 00:00:24:00:00:00:00:00
40 00:00:00:00:00:00:00:00
41 00:00:1f:00:00:00:00:00
42 00:00:00:00:00:00:00:00
43 00:00:27:00:00:00:00:00
```

[https://blog.csdn.net/qq\\_44108455](https://blog.csdn.net/qq_44108455)

写个python脚本来提取鼠标移动坐标:

keyboard1.py

```

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>", "4
3":"<F10>", "44":"<F11>", "45":"<F12>"}
output = []
keys = open('usbdata.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 li
ne[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))

```

运行得到如下结果:



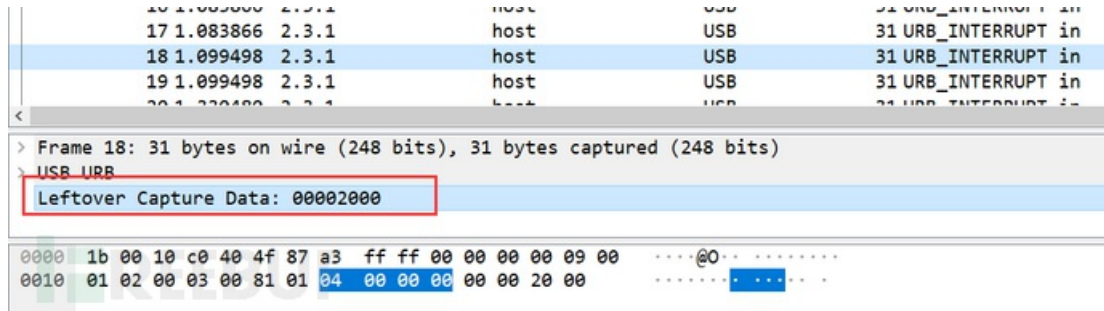
7200[DEL]53[DEL]93

因为[DEL]是删除键, 所以, flag是720593

## 002鼠标流量

USB协议鼠标数据部分在Leftover Capture Data域中, 数据长度为四个字节。

其中第一个字节代表按键, 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。第二个字节可以看成是一个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。第三个字节与第二字节类似, 代表垂直上下移动的偏移。数据如下图所示:



如上图所示数据信息为0x00002000, 表示鼠标垂直向上移动20。

题型:

Flag藏于usb流量中, 通过USB协议数据中的鼠标移动轨迹转换成Flag。

解题思路:

使用kali linux中的tshark 命令把cap data提取出来: `tshark -r usb.pcap -T fields -e usb.capdata > usbdata.txt`,

并去除空行 `tshark -r usb.pcapng -T fields -e usb.capdata | sed '/^\s*$/d' > usbdata.txt`。

根据usb协议鼠标数据还原鼠标移动轨迹。

练练手:

这是一道鼠标流量分析题。

方法一:

先用 tshark 命令把cap data提取出来

```
tshark -r usb.pcap -T fields -e usb.capdata > usbdata.txt
```

每次需要变得就是usb.pcap这里

接下来使用脚本把坐标显示出来, python脚本:

```

nums = []
keys = open('usb.txt','r')
posx = 0
posy = 0
for line in keys:
    if len(line) != 12 :
        continue
    x = int(line[3:5],16)
    y = int(line[6:8],16)
    if x > 127 :
        x -= 256
    if y > 127 :
        y -= 256
    posx += x
    posy += y
    btn_flag = int(line[0:2],16) # 1 for left , 2 for right , 0 for nothing
    if btn_flag == 1 :
        print posx , posy
keys.close()

```

```

root@kali:~/llfx/mouse# python mouse.py > xy.txt
root@kali:~/llfx/mouse# cat xy.txt
-92 416
498 -183
-232 -353
-233 -352
-233 -351
-233 -350
-233 -348
-233 -346
-233 -343
-233 -341
-233 -336

```

[https://blog.csdn.net/qq\\_44108455](https://blog.csdn.net/qq_44108455)

接下来使用画图工具，gnuplot

如果没有下载，可以先下载：`apt install gnuplot`

然后输入：

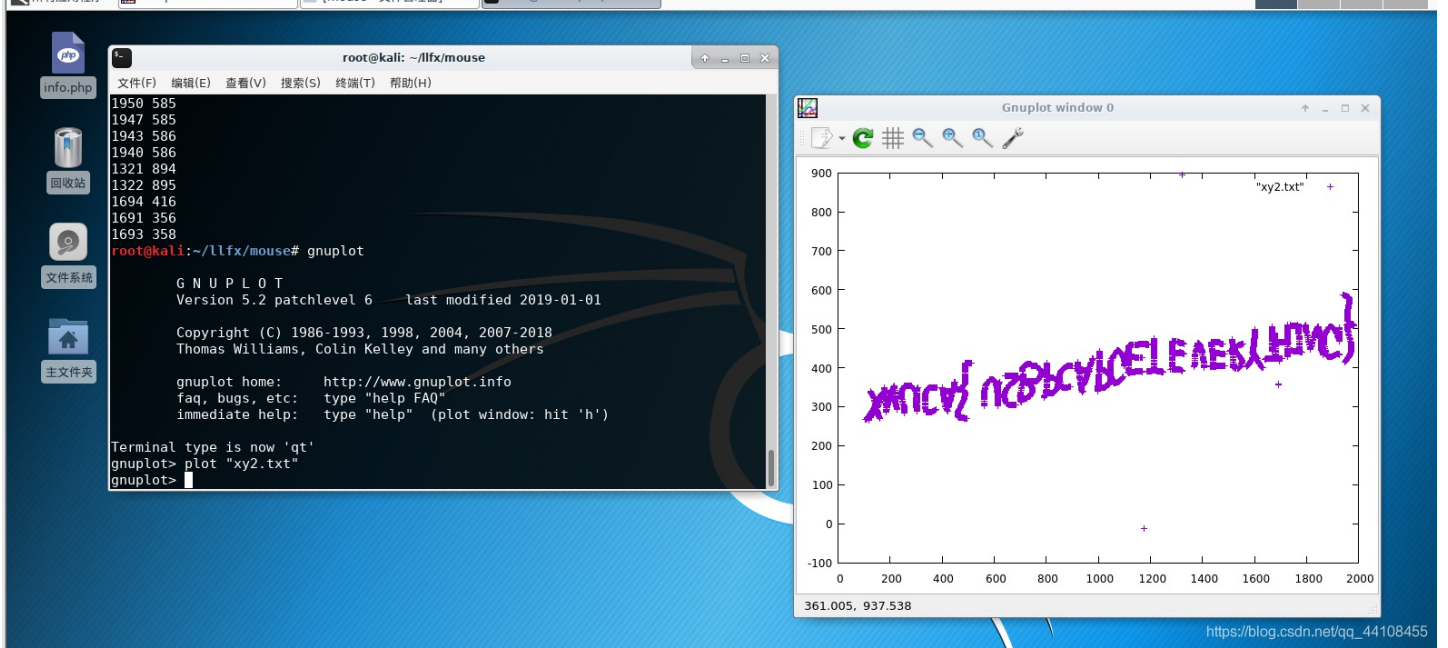
```

gnuplot
plot "xy.txt"

```

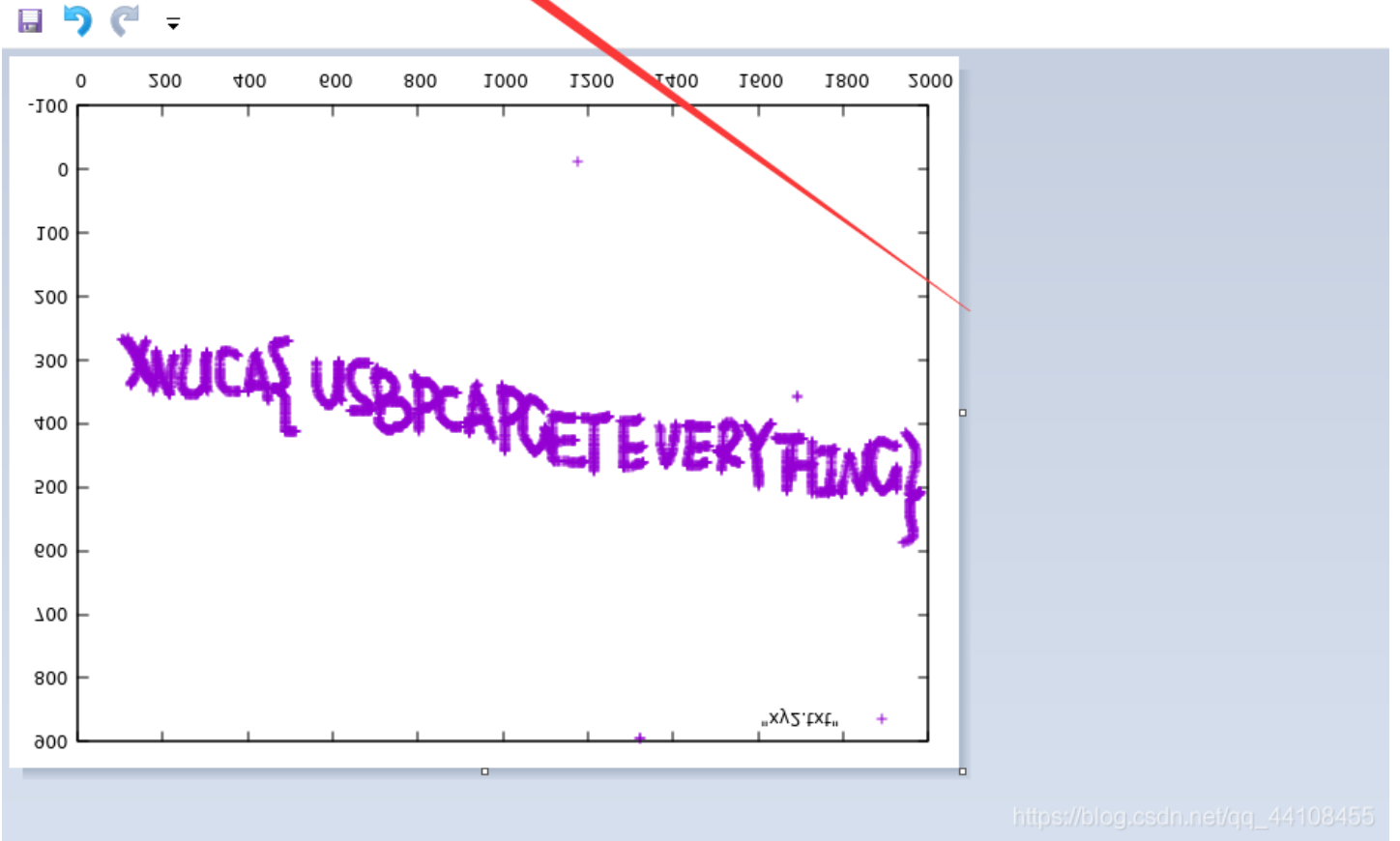
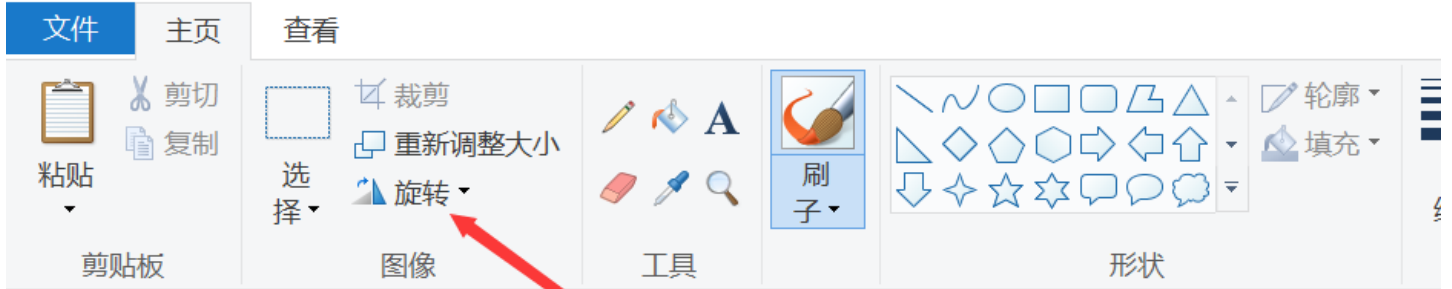


执行之后发现这个图很奇怪，说明上面mouse.py中的代表左键或者右键的那个式子需要改变一下，改成2之后重新运行



okk，用画图工具垂直翻转一下就可以啦！

flag.png - 画图



### 方法二:

使用王一航大佬的脚本然后直接运行就可以

[https://blog.csdn.net/qq\\_44108455](https://blog.csdn.net/qq_44108455)

[https://blog.csdn.net/qq\\_44108455](https://blog.csdn.net/qq_44108455)

```
python UsbMiceDataHacker.py usb2.pcap RIGHT
```

## 参考链接:

[CTF流量分析之题型深度解析](#)

[第一届“百度杯”信息安全攻防总决赛 线上选拔赛](#)

<https://www.cnblogs.com/sn1per/p/12553064.html>



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)