ctf基本操作

原创

mafucan 于 2020-06-21 15:44:28 发布

CTF在线演练

当具备了一定的知识储备后就需要在实际的环境刷题对知识进行巩固，而且在刷题的过程中也会学习到新的知识点，总结解题思路，开启脑洞。

需要下载镜像去 i tell you下载ISO源文件

文件操作与隐写

文件类型识别

kali里的File命令 file + 文件名，确定文件类型

winhex查看文件头类型，根据头文件类型判断

文件类型对应不同的文件头

notepad++在插件找到16进制内容查看，寻找对应的内容（经常性的可以查看16进制，然后转换）

文件头残缺/错误

先用file查看猜测大概的类型，无显示为data，考虑进行填补

3.文件头残缺/错误

通常文件无法正常打开有两种情况，一种是文件头部残缺，另一种是文件头部字段错误。针对文件头部残缺的情况，使用winhex程序添加相应的文件头，针对头部字段错误，可以找一个相同类型的文件进行替换。

使用场景：文件头部残缺或文件头部字段错误无法打开正常文件。

格式：file 文件名

```
root@kali2:~/ctf# file stef.png
stef.png: data
root@kali2:~/ctf# file misc100f.zip
misc100f.zip: data
```

缺失进行补头

binwalk工具

用法：

分析文件：binwalk filename

分离文件: binwalk -e filename

foremost

foremost 文件名 -o 输出目录名（后面的-o其实可以不用）

dd

当文件自动分离出错或者因为其他原因无法自动分离时，可以使用dd实现文件手动分离。

格式：

dd if=源文件 of=目标文件名名 bs=1 skip=开始分离的字节数

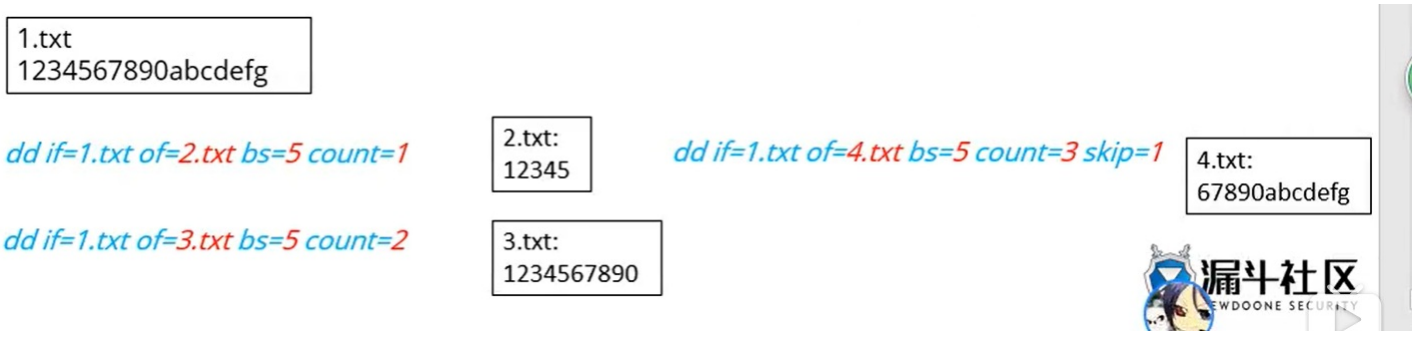
参数说明：

if=file　　#输入文件名，缺省为标准输入。
of=file　　#输出文件名，缺省为标准输出。
bs=bytes　#同时设置读写块的大小为bytes，可代替ibs和obs。
skip=blocks #从输入文件开头跳过blocks个块后再开始复制。

未命名• - Typora

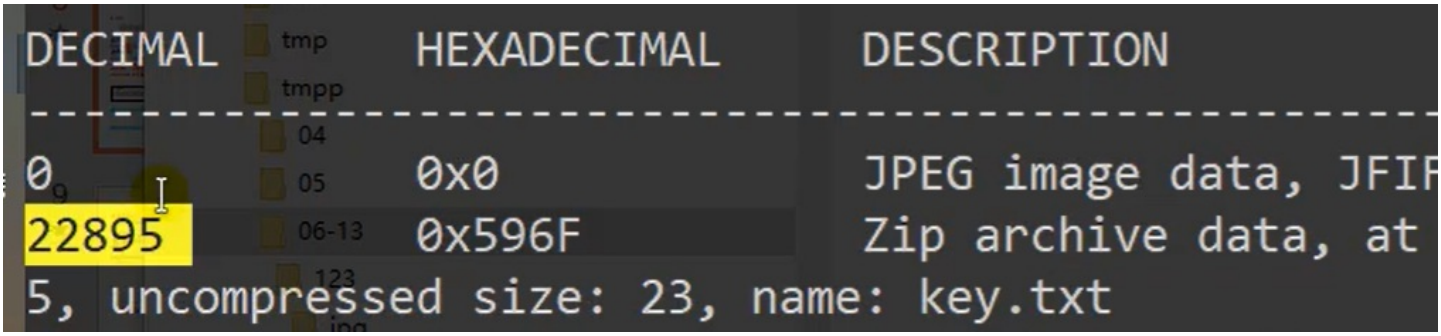
手动进行分离

dd if=源文件 of=目标文件名 bs=1 skip=开始分离的字节数



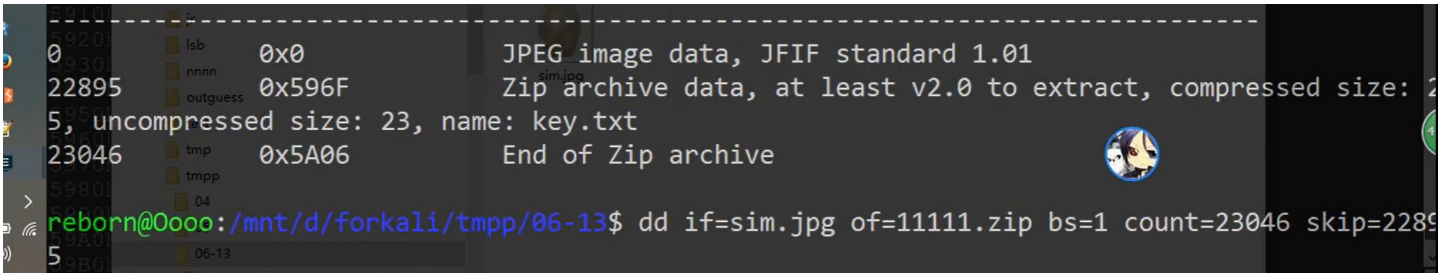
skip开始分离，就是跳过前面的内容，选取后面的部分

bs读取的大小，5代表取读5位，count为这样的bs的数目，如果加入skip，就是对应跳过bs----bs为5位



decimal：十进制，hexadecimal：十六进制

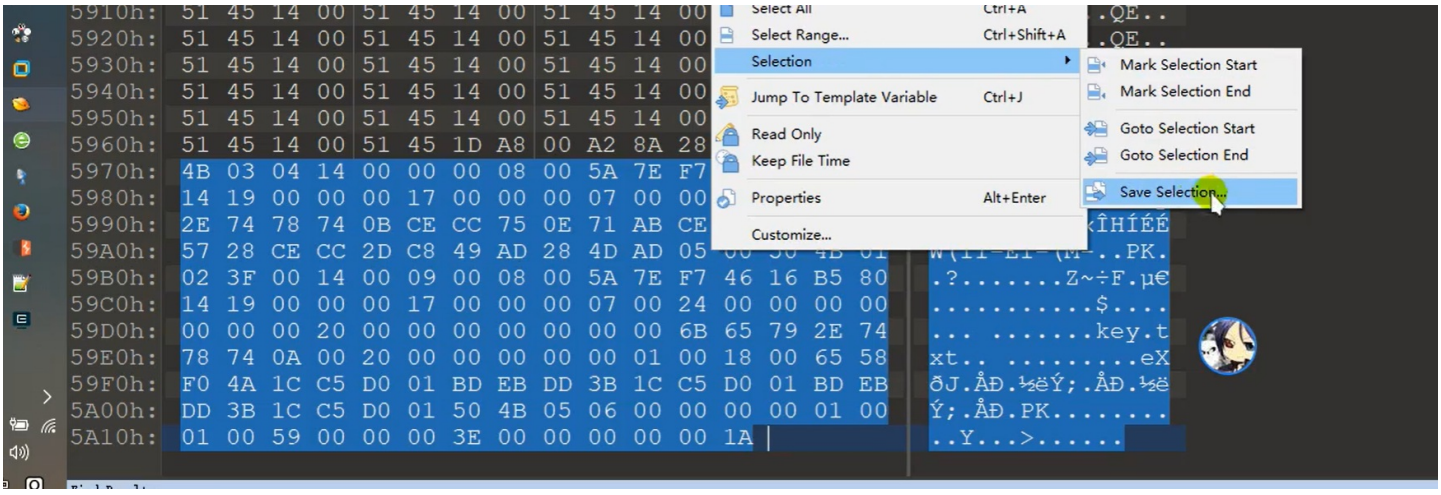
可以找到对应的在winhex之类里面的位置进行验证



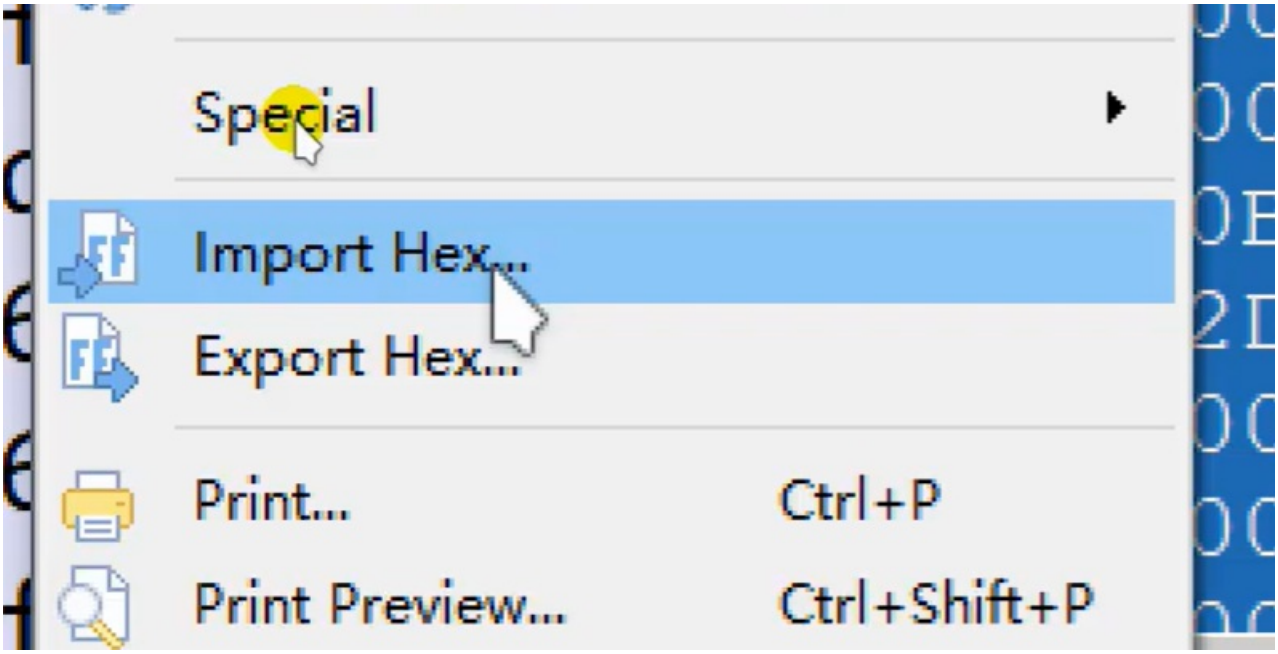
取值zip，是从22895-23046，选择前面数一共23046位，skip跳过前面22895位，直接得到zip文件

010Editor

在binwalk后知道对应的zip为后面的多少位，找到16进制中对应的位置，在这里直接右键，进行选择，保存选择，保存为zip，直接出结果



发现一些题型，打开为一串16进制，打开010editor选择进行导入，再输出为对应的文件类型



文件合并操作

1.Linux下的文件合并

使用场景：linux下,通常对文件名相似的文件要进行批量合并

格式： cat 合并的文件 > 输出的文件

```
root@kali2:~/ctf/cat# cat chapter01 chapter02 chapter03 > book
root@kali2:~/ctf/cat# cat chapter* > book1
```

完整性检测：linux 下计算文件md5：

md5sum *文件名*

```
reborn@Oooo:/mnt/d/forkali$ md5sum sim.jpg
d09e8a07b6dedb0633aa3c432f931362  sim.jpg
```

2.Windows下的文件合并

使用场景：windows下，通常要对文件名相似的文件进行批量合并

格式： copy /B 合并的文件 输出的文件命令

```
D:\CTF\copy>copy /B chapter01+chapter02+chapter03 book
chapter01
chapter02
chapter03
已复制         1 个文件。

D:\CTF\copy>copy /B chapter* book1
chapter01
chapter02
chapter03
已复制         1 个文件。
```

完整性检测：windows下计算文件md5：

certutil -hashfile *文件名* md5

```
reborn@OOOO D:\forkali
# certutil -hashfile sim.jpg md5
MD5 的 sim.jpg 哈希:
d09e8a07b6dedb0633aa3c432f931362
CertUtil: -hashfile 命令成功完成。
```

cat gif01 gif02 >1.gif

生成新的gif，结合完成，可能还要添加头文件

文件内容隐写

直接搜索查找目标如：key、flag

可以在左下角find Hex Bytes 那里寻找字符串

图片隐写

>> 图片隐写的常见隐写方法

1. 细微的颜色差别
2. GIF图多帧隐藏
    1. *颜色通道隐藏*
    2. *不同帧图信息隐藏*
    3. *不同帧对比隐写*
3. Exif信息隐藏
4. 图片修复
    1. *图片头修复*
    2. *图片尾修复*
    3. *CRC校验修复*
    4. *长、宽、高度修复*
5. 最低有效位LSB隐写
6. 图片加密
    1. *Stegdetect*
    2. *outguess*
    3. *Jphide*
    4. *F5*

1.Firework（某绘图工具）

查看隐写的图片文件，通过firework可以找到隐藏图片(找图和层)

2.Exif

Google地球，查看gps位置

Linux操作：exiftool 1.jpg

3.StegSlove

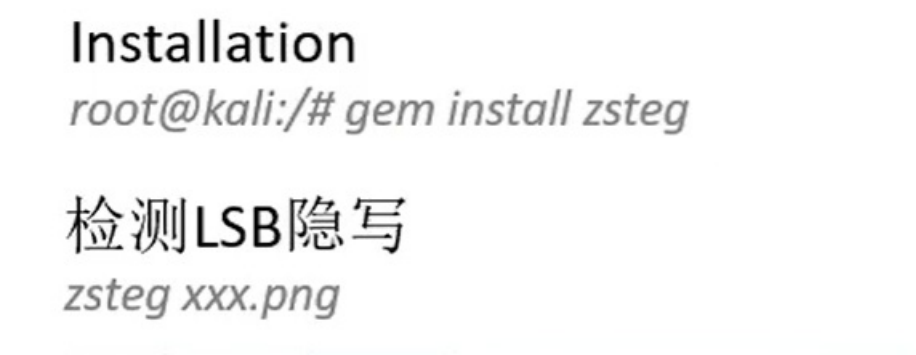
analyse最后的是不同的图片进行对比（异或、添加），多在两张大概相同的图片进行对比

Image Combiner:拼图，图片拼接

在这里面的减操作，是不同的，互相的减不同

4.LSB

zsteg工具



显示所有可能的文本信息，防止过多的排序

wbstego4：解密通过lsb加密的图片

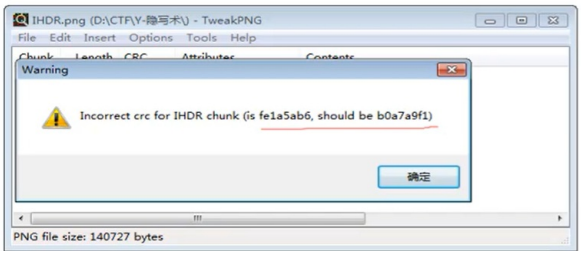
用画图进行图片格式转换

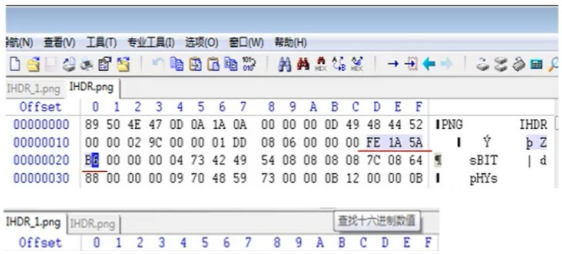
python脚本自己来写

5.TweakPNG

使用场景：文件头正常却无法打开文件，利用TweakPNG修改CRC

1.当PNG文件头正常但无法打开文件，可能是CRC校验出错，可以尝试通过TweakPNG打开PNG，会弹出校验错误的提示，这里显示CRC是fe1a5ab6，正确的是b0a7a9f1。打开winhex搜索fe1a5ab6将其改为b0a7a9f1。






```python
import os
import binascii
import struct
crcbp = open("2.png","rb").read()
for i in range(1024):
    for j in range(1024):
        data = crcbp[12:16] + struct.pack('>i',i) + struct.pack('>i',j) +
        crcbp[24:29]
        crc32 = binascii.crc32(data) & 0xffffffff
        if crc32 == 0xcbd6df8a:
            print i,j
            print "hex",hex(i),hex(j)
```

python脚本修改

6.Bftools

在windoes的cmd下，对加密过的图片文件进行解密

格式：

Bftools.exe decode braincopter 要解密的图片名称–output 输出文件名

Bftools.exe run 上一步输出的文件

```
D:\CTF\bftools\bftools>bftools.exe decode braincopter zzzzzzyu.png --output 123.
png

D:\CTF\bftools\bftools>bftools.exe run 123.png
XDCTF{ji910-dad9jq0-iopuno}
D:\CTF\bftools\bftools>
```

7.SilentEye

silenteye是一款可以将文字或者文件隐藏到图片的解密工具。

使用场景：windows下打开silentEye工具，对加密的图片进行解密

例：

1.使用silentEye程序打开目标图片，点击image->decode，点击decode，可以查看隐藏文件，点击保存即可


8.jpg图像加密

Stegdetect 工具探测加密方式

Stegdetect程序主要用于分析JPEG文件。因此用Stegdetect可以检测到通过JSteg、JPHide、OutGuess、Invisible Secrets、F5、appendX和Camouflage等这些隐写工具隐藏的信息。

stegdetect xxx.jpg
stegdetect –s *敏感度 xxx.jpgexi*

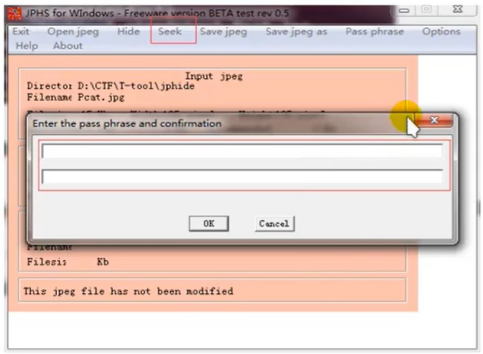
```
thinking@ubuntu:~/Desktop$ stegdetect 123456.jpg
123456.jpg : f5(***)
thinking@ubuntu:~/Desktop$ stegdetect angrybird.jpg
angrybird.jpg : outguess(old)(*)
thinking@ubuntu:~/Desktop$ stegdetect Pcat.jpg
Pcat.jpg : negative
thinking@ubuntu:~/Desktop$ stegdetect -s 10.0 Pcat.jpg
Pcat.jpg : jphide(*)
thinking@ubuntu:~/Desktop$
```

2)Jphide

Jphide是基于最低有效位LSB的JPEG格式图像隐写算法.

例：

Stegdetect提示jphide加密时，可以用Jphs工具进行解密，打开jphswin.exe，使用open jpeg打开图片，点击seek，输入密码和确认密码，在弹出文件框中选择要保存的解密文件位置即可，结果保存成txt文件。



3) Outguess

outguess一般用于解密文件信息。

使用场景：Stegdetect识别出来或者题目提示是outguess加密的图片

该工具需编译使用：./configure && make && make install

格式：outguess –r *要解密的文件名 输出结果文件名*

```
root@kali2:~/ctf# outguess -r angrybird.jpg angry.txt
Reading angrybird.jpg....
Extracting usable bits:   36252 bits
Steg retrieve: seed: 152, len: 14
root@kali2:~/ctf# cat angry.txt
flag{Out_Gas}
```

4) F5

F5一般用于解密文件信息。

使用场景： Stegdetect识别出来是F5加密的图片或题目提示是F5加密的图片

进入F5-steganography_F5目录，将图片文件拷贝至该目录下，从CMD进入该目录

格式：Java Exrtact 要解密的*文件名*–p *密码*

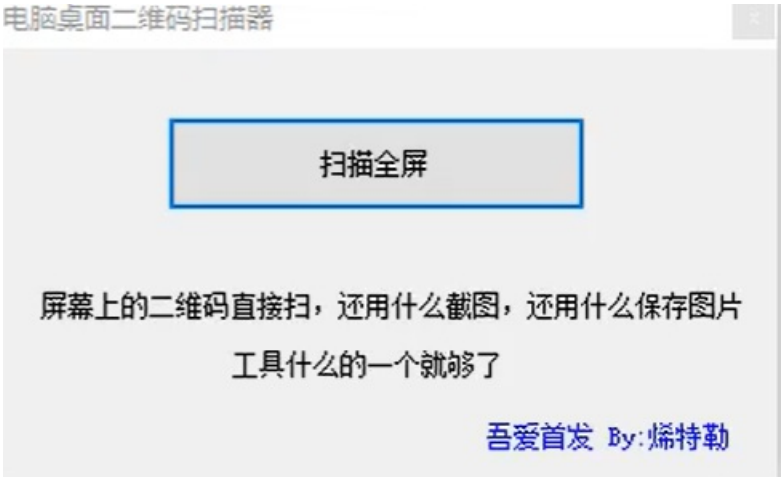
```
D:\CTF\T-tool\F5-steganography-master_F5>java Extract 123456.jpg -p 123456
Huffman decoding starts
Permutation starts
614400 indices shuffled
Extraction starts
Length of embedded file: 20 bytes
(1, 127, 7) code used
```

运行结束后我们可以直接在目录下的output.txt中看到结果。

9.二维码处理

CQR.exe

取反，进行反色操作



一个蛮厉害的工具，要去找找

压缩文件处理

1.伪加密

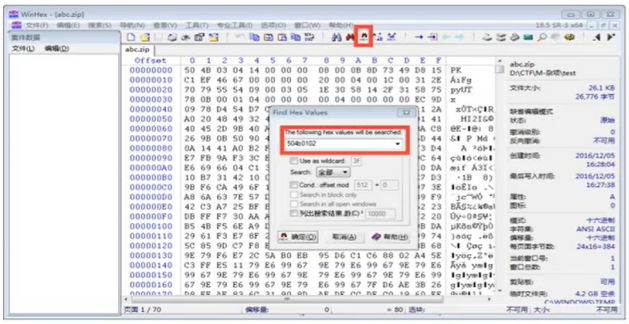
1.伪加密

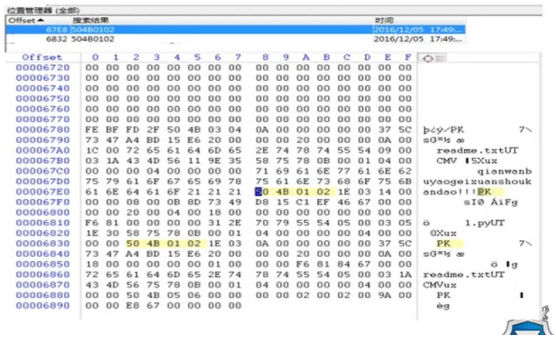
如果压缩文件是加密的，或文件头正常但解压缩错误，首先尝试文件是否为伪加密。zip文件是否加密是通过标识符来显示的，在每个文件的文件目录字段有一位专门标识了文件是否加密，将其设置为00表示该文件未加密，如果成功解压则表示文件为伪加密，如果解压出错说明文件为真加密。

使用场景：伪加密文件

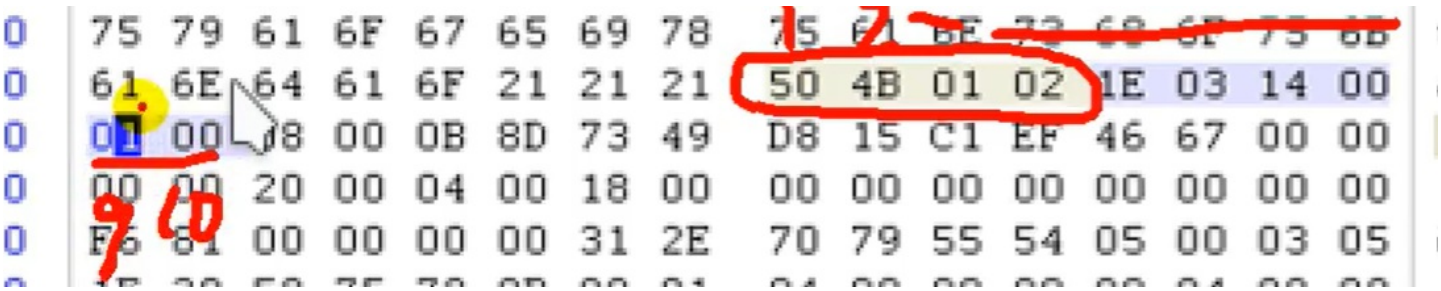
操作方法：使用winhex打开压缩文件，找到文件头第九第十个字符，将其修改为0000。

1.使用winhex打开文件搜索16进制504B0102，可以看到每个加密文件的文件头字段。





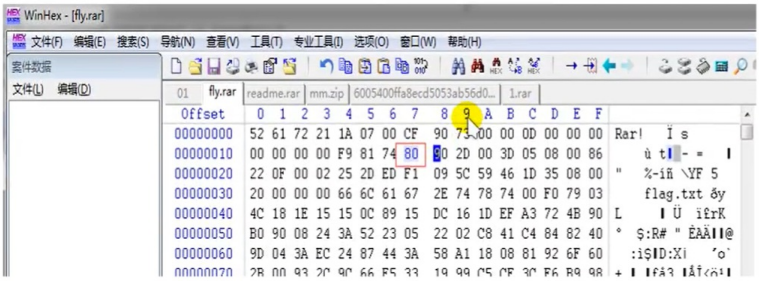
360解压，直接进行解压，先找到504B0101，再往后数到第九第十个字符（两位数字一字符）50也算入



注意位数

RAR文件

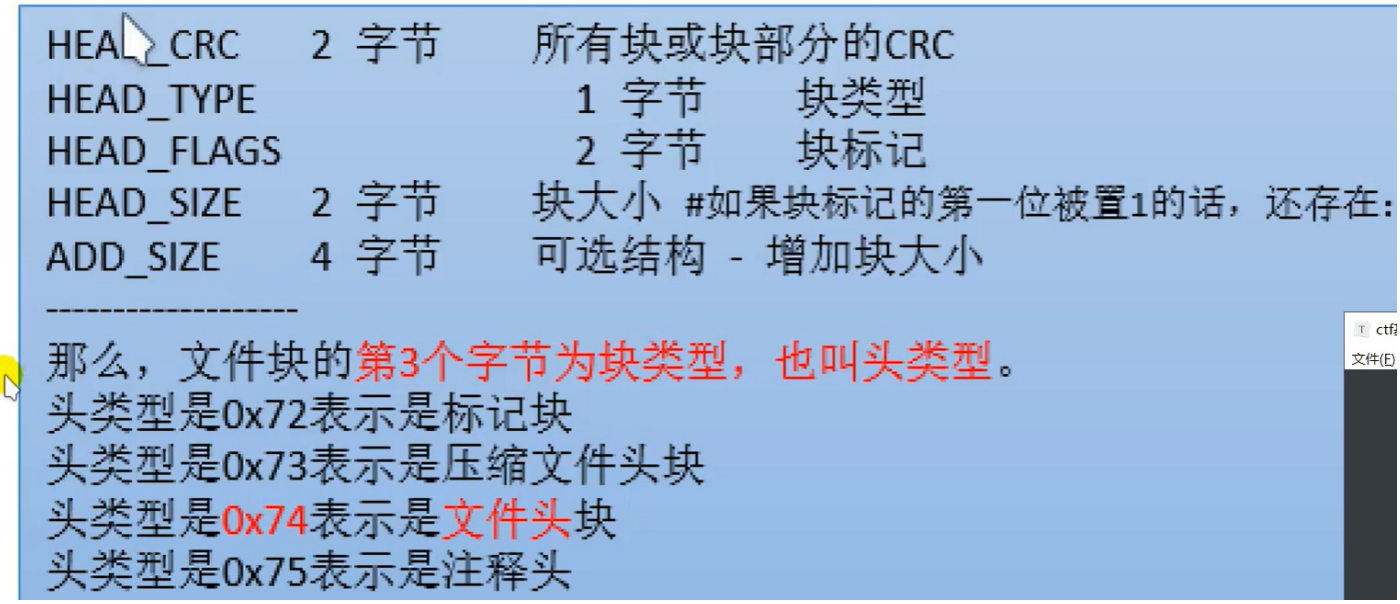
3. RAR文件由于有头部校验，使用伪加密时打开文件会出现报错，使用winhex修改标志位后如报错消失且正常解压缩，说明是伪加密。使用winhex打开RAR文件，找到第24个字节，该字节尾数为4表示加密，0表示无加密，将尾数改为0即可破解伪加密。



明文攻击，后面未知的位数添加为？？？

加密密钥也可能为flag

压缩软件，可以选择不同的压缩算法



改为0x74，添加头文件解压出来，数三位，先找到前面的结尾。最后再这里修改正确的头文件

流量取证（磁盘、内存取证）

流量包文件分析

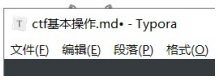
CTF 比赛中，**流量包的取证分析**是另一项重要的考察方向。

通常比赛中会提供一个包含流量数据的PCAP 文件，有时候也会需要选手们先进行修复或重构传输文件后，再进行分析。

- **总体把握**
  - 协议分级
  - 端点统计
- **过滤筛选**
  - 过滤语法
  - Host，Protocol，contains，特征值
- **发现异常**
  - 特殊字符串
  - 协议某字段
  - flag 位于服务器中
- **数据提取**
  - 字符串取
  - 文件提取

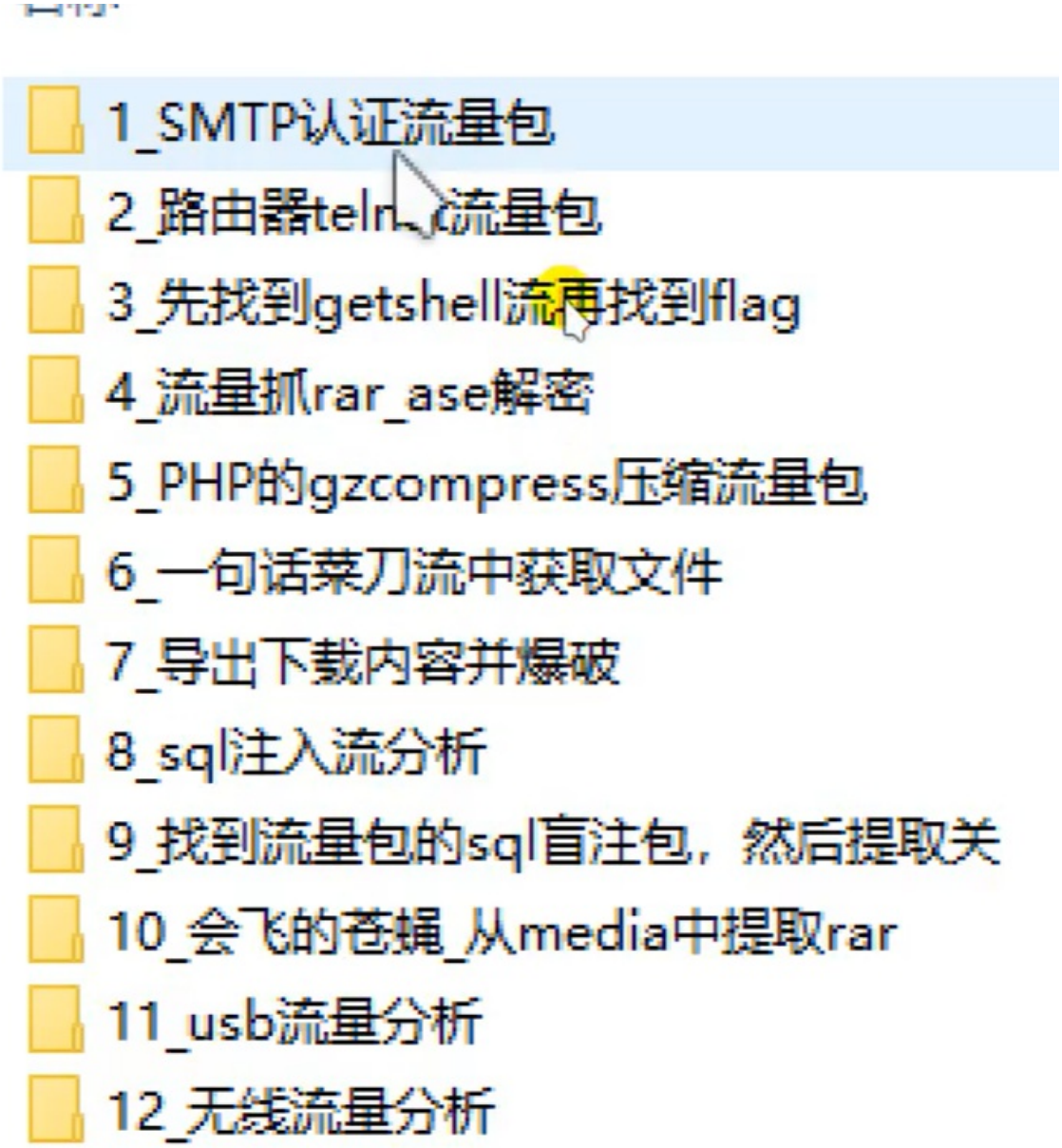
总的来说比赛中的流量分析可以概括为以下三个方向:

- 流量包修复
- 协议分析
- 数据提取



找http（协议分级）

wireshark工具的使用



12道大概的题

截握手包，破取WiFi密码

wireshark过滤器

Source	Destination
192.168.1.102	115.231.236.116
115.231.236.116	192.168.1.102
192.168.1.102	115.231.236.116

源、目的地

1.过滤IP，如源ip或者mubiaox.x.x.x.

常用的过滤命令：

1.过滤IP，如源IP或者目标 x.x.x.x

```
ip.src eq x.x.x.x or ip.dst eq x.x.x.x 或者 ip.addr eq x.x.x.x
```

2.过滤端口

```
tcp.port eq 80 or udp.port eq 80
tcp.dstport == 80  只显tcp协议的目标端口为80
tcp.srcport == 80  只显tcp协议的源端口为80
tcp.port >= 1 and tcp.port <= 80
```

协议和对应的端口

3.过滤协议

```
tcp/udp/arp/icmp/http/ftp/dns/ip……
```

4.过滤MAC

```
eth.dst == A0:00:00:04:C5:84 过滤目标mac
```

5.包长度过滤

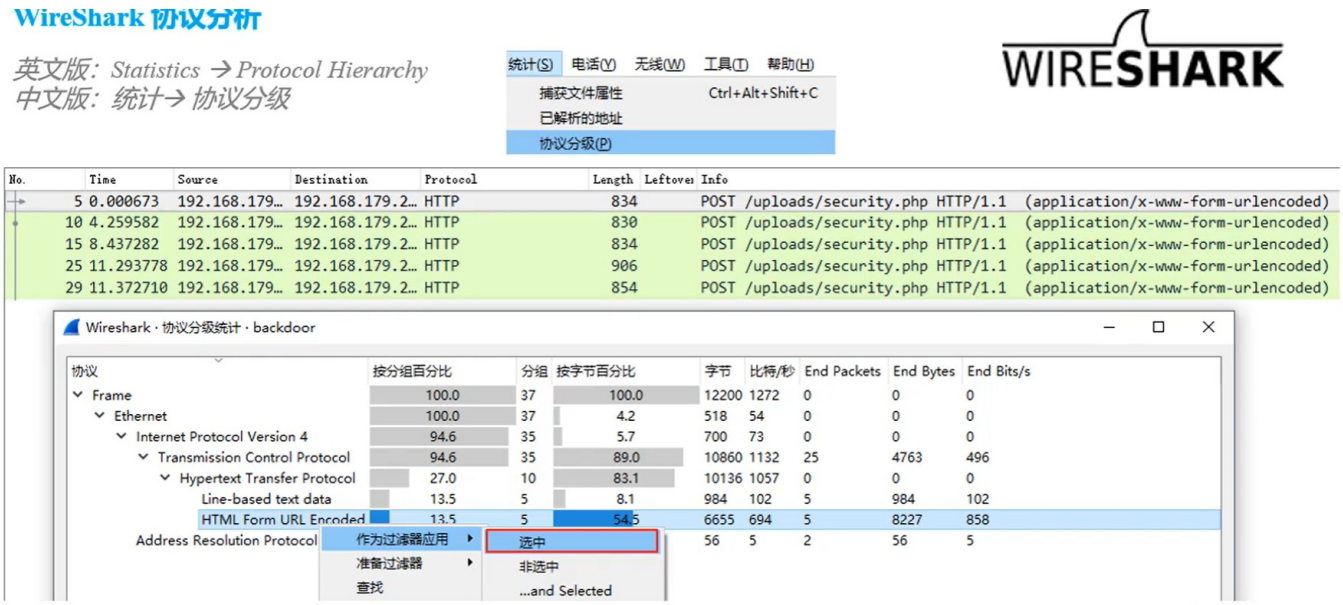
```
udp.length == 26 这个长度是指udp本身固定长度8加上udp下面那块数据包之和 。
tcp.len >= 7 指的是ip数据包(tcp下面那块数据),不包括tcp本身
ip.len == 94 除了以太网头固定长度14,其它都算是ip.len,即从ip本身到最后
frame.len == 119 整个数据包长度,从eth开始到最后
```

6.http模式过滤

```
http.request.method == "GET"
http.request.method == "POST"
http.request.uri == "/img/logo-edu.gif"
http contains "GET"
http contains "HTTP/1."
http.request.method == "GET" && http contains "User-Agent:"
http contains "flag"
http contains "key"
tcp contains "flag"
```

看http，再搜索关键词

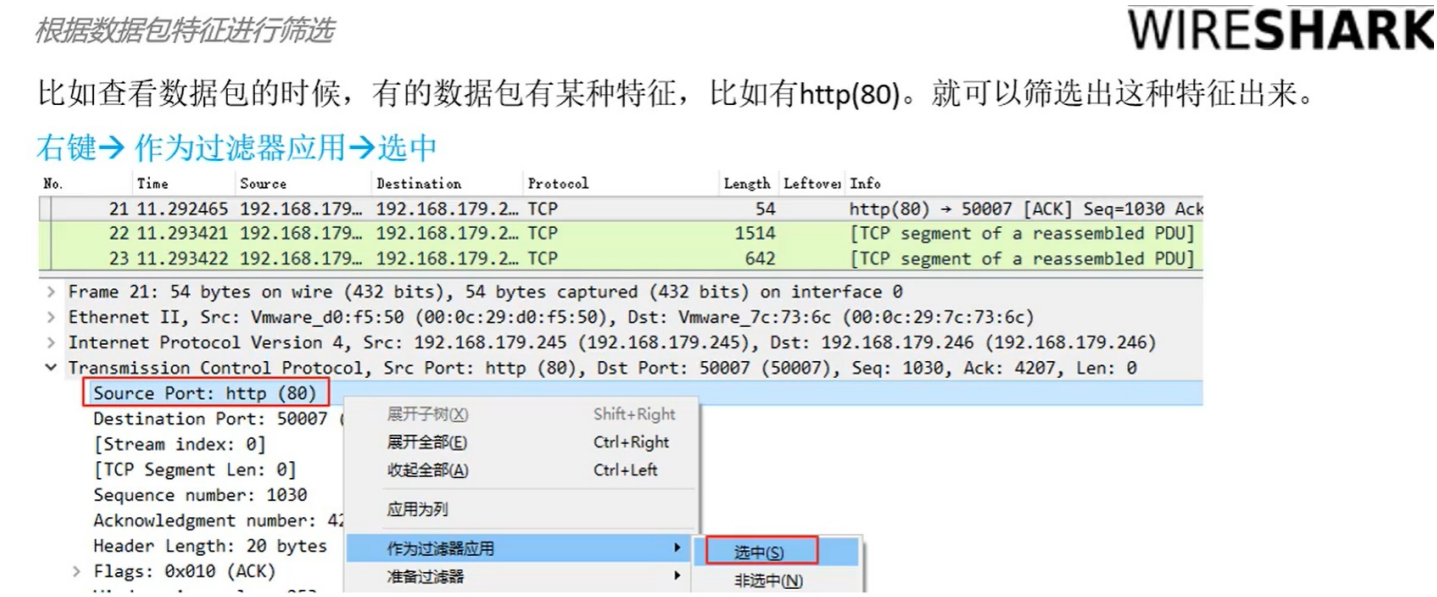
contains选择是否包含这个字段



协议分析

统计-协议分级

去找流量包的大概分区

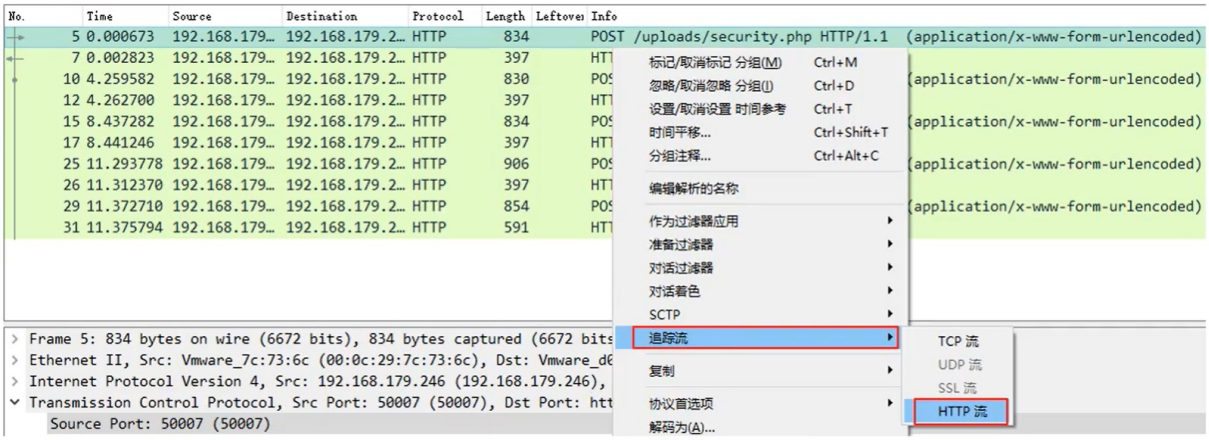


流汇聚

在关注的http数据包或tcp数据包中选择流汇聚，可以将HTTP流或TCP流汇聚或还原成数据，在弹出的框中可以看到数据内容。

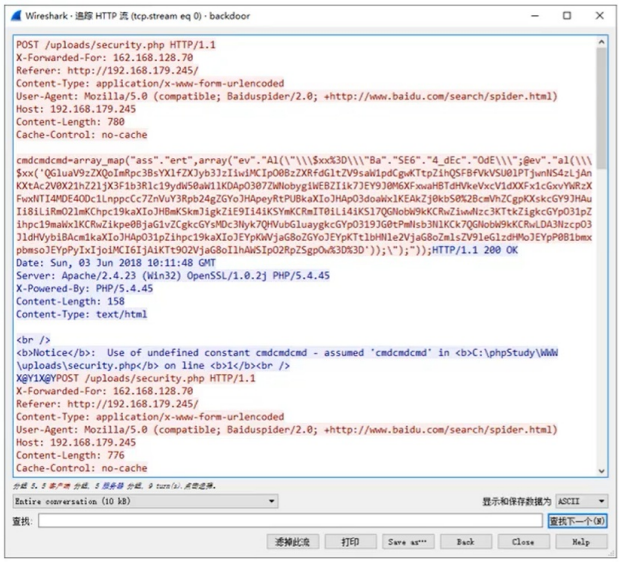
WIRESHARK

HTTP流：



追踪流，进行搜索flag

在关注的http数据包或tcp数据包中选择流汇聚，可以将HTTP流或TCP流汇聚或还原成数据，在弹出的框中可以看到数据内容。



常见的HTTP流关键内容：

- 1、HTML中直接包含重要信息。
- 2、上传或下载文件内容，通常包含文件名、hash值等关键信息，常用POST请求上传。
- 3、一句话木马，POST请求，内容包含eval，内容使用base64加密

一句话木马，进行POST请求，直接连接后门

命令行的操作方式shell

搜索样例，一般是先http，然后再tcp

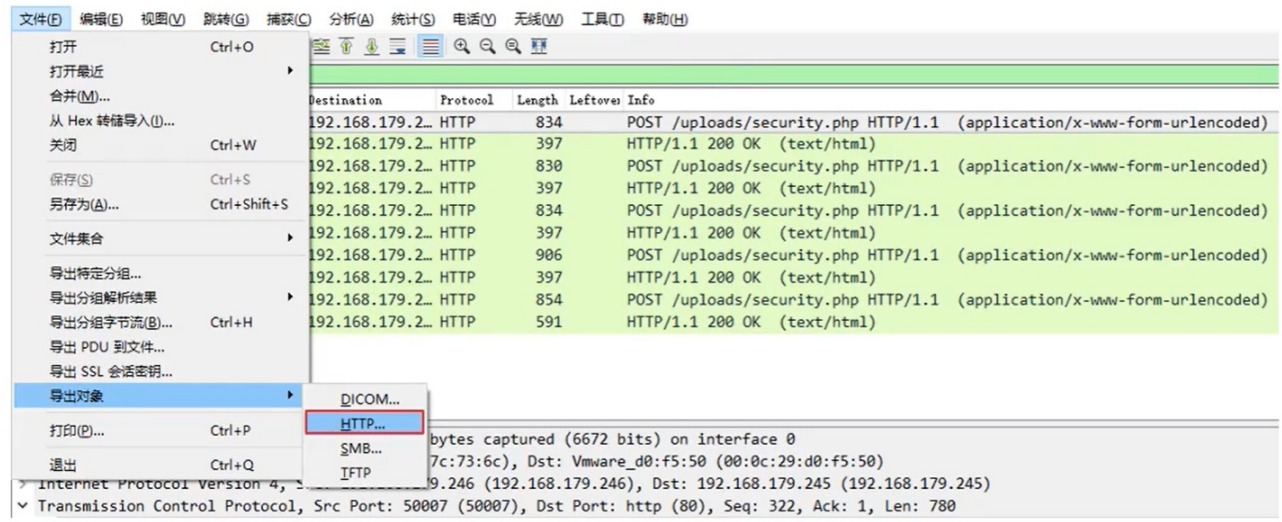
技巧：直接调取contains函数去搜索flag

wireshark数据提权

WireShark 数据提取

1. 使用wireshark可以自动提取通过http传输的文件内容。

文件→ 导出对象→ HTTP



寻找导出对象，导出后看十六进制的头文件猜测大概的属性，导出到文件夹，改名，导出

2. wireshark可以手动提取文件内容。

点击想要的数据包，如下图 选定media type 的位置
右键 → 导出分组字节流 或者 点击菜单栏 文件→ 导出分组字节流，快捷方式Ctrl+H
在弹出的框中将文件保存成二进制文件。

找大小，偏大的导出

无线wifi流量包

应用显示过滤器 … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Leftover	Info
1	0.000000	Tp-LinkT_5d:d0:ee	Broadcast	802.11	143		Beacon frame, SN=6, FN=0, Flags=........
2	-0.000042		Tp-LinkT_5d:d0:ee (00:1d:0f:5d:d0:ee) (RA)	802.11	10		Clear-to-send, Flags=........
3	0.000002	Tp-LinkT_5d:d0:ee	HuaweiTe_f8:4c:54	802.11	137		Probe Response, SN=8, FN=0, Flags=.....
4	-0.000034		Tp-LinkT_d9:49:7e (5c:63:bf:d9:49:7e) (RA)	802.11	10		Acknowledgement, Flags=........
5	0.000000	Tp-LinkT_5d:d0:ee	HuaweiTe_f8:4c:54	802.11	137		Probe Response, SN=8, FN=0, Flags=....R.
6	-0.000034	Apple_4c:2a:9a (98:f…	Tp-LinkT_d9:49:7e (5c:63:bf:d9:49:7e) (RA)	802.11	16		Request-to-send, Flags=........
7	-0.000028		Apple_4c:2a:9a (98:fe:94:4c:2a:9a) (RA)	802.11	10		Clear-to-send, Flags=........

协议分析发现只有wireless LAN协议，很有可能是WPA或者WEP加密的无线数据包。

Wireshark · 协议分级统计 · shipin

协议	按分组百分比	分组	按字节百分比	字节	比特/秒	End Packets	End Bytes	End Bits/s
Frame	100.0	16664	100.0	292214	62 k	0	0	0
IEEE 802.11 wireless LAN	100.0	16664	77.9	227602	48 k	15826	207490	44 k
Logical-Link Control	0.0	3	0.1	423	89	0	0	0
802.1X Authentication	0.0	3	0.1	399	84	3	399	84
IEEE 802.11 wireless LAN management frame	4.8	802	1.4	4029	855	802	4029	855
Data	0.2	33	1.1	3276	695	33	3276	695

无显示过滤器.

Close 复制 Help

无线WiFi流量包

aircrack-ng 工具进行wifi 密码破解

1. 用aircrack-ng检查cap包：*aircrack-ng xxx.cap*

```
root@kali:~/Desktop# aircrack-ng shipin.cap
Opening shipin.cap
Read 16664 packets.

   #  BSSID              ESSID                     Encryption

   1  00:1D:0F:5D:D0:EE  0719                      WPA (1 handshake)

Choosing first network as target.

Opening shipin.cap
Please specify a dictionary (option -w).

Quitting aircrack-ng...
```

2. 用aircrack-ng跑字典进行握手包破解：*aircrack-ng xxx.cap –w pass.txt*

```
root@kali:~/Desktop# aircrack-ng shipin.cap -w pass.txt
Opening shipin.cap
Read 16664 packets.

   #  BSSID              ESSID                     Encryption

   1  00:1D:0F:5D:D0:EE  0719                      WPA (1 handshake)

Choosing first network as target.

Opening shipin.cap
Reading packets, please wait...

                                Aircrack-ng 1.2 rc4

      [00:00:00] 8/2492 keys tested (296.77 k/s)

      Time left: 8 seconds                                      0.32%

                           KEY FOUND! [ 88888888 ]


      Master Key     : B4 30 38 0F 24 7B 57 AC DE B5 3A 7F 2E FE 6B 45
                       0B 34 02 C3 89 F9 69 D5 B7 35 87 1B FB 4C EE 7F

      Transient Key  : 17 AE 23 D0 69 7C 0D 45 2B 40 F6 7D 06 C9 C5 6F
                       25 F0 B0 48 7A 6C 22 7C E2 73 50 71 46 FE 5D 0C
                       8F 59 01 BE 66 56 DF 1E 58 DD 34 DB BF A7 2D FD
                       2C 53 11 7F B2 E5 F0 16 7F 57 F5 6A 04 36 F5 71

      EAPOL HMAC     : 75 19 C5 F3 3E 33 58 23 CA 4B A1 85 FB 46 C0 2A
```

1.用aircrack-ng检查cap包 ： aircrack-ng xxx.cap

handshake 握手包

bssid :WiFi地址

essid: WiFi名字

跑字典破解WiFi密码

aircrack-ng xxx.cap -w +字典

USB流量

USB流量也是流量分析题的考查点，一般考察的流量涉及键盘击键，鼠标移动与点击，存储设备的明文传输通信，USB无线网卡网络传输内容等。

USB协议的数据部分在Leftover Capture Data域之中。

右键leftover capture data –> 应用为列。

No.	Time	Source	Destination	Protocol	Length	Leftover Capture Data	Info
3	0.520044	3.10.1	host	USB	35	00fe0000feff0000	URB_INTERRUPT in
953	54.399801	3.10.1	host	USB	35	00fdff00fdffffff	URB_INTERRUPT in
947	54.351857	3.10.1	host	USB	35	00fdff00fdffffff	URB_INTERRUPT in
854	49.111879	3.10.1	host	USB	35	00fdff00fdffffff	URB_INTERRUPT in
691	36.423471	3.10.1	host	USB	35	00fdff00fdffffff	URB_INTERRUPT in

```
> Frame 3: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)
> USB URB
  Leftover Capture Data: 00fe0000feff0000
```

```
0000  1b 00 f0 8a 42 83 03 97  ff ff 00 00 00
0010  01 03 00 0a 00 81 01 08  00 00 00 00 fe
0020  ff 00 00
```

展开子树(X) Shift+Right
展开全部(E) Ctrl+Right
收起全部(A) Ctrl+Left
应用为列

可以将该域的值在主面板上显示，键盘数据包的数据长度为**8个字节**，击键信息集中在第3个字节，每次key stroke都会产生一个keyboard event usb packet 。

USB流量包文件分析（一定要先应用为列）

可以获取鼠标的流量曲线

查看第三个字节对应的值，去官方手册查看对应的情况和数据

USB键盘流量抓取分析

Leftover Capture Data中值与具体键位的对应关系，可以参考：
http://www.usb.org/developers/hidpage/Hut1_12v2.pdf

Table 12: Keyboard/Keypad Page

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
0	00	Reserved (no event indicated)[9]	N/A	√	√	√	4/101/104
1	01	Keyboard ErrorRollOver[9]	N/A	√	√	√	4/101/104
2	02	Keyboard POSTFail[9]	N/A	√	√	√	4/101/104
3	03	Keyboard ErrorUndefined	N/A	√	√	√	4/101/104
4	04	Keyboard a and A[4]	31	√	√	√	4/101/104
5	05	Keyboard b and B	50	√	√	√	4/101/104
6	06	Keyboard c and C[4]	48	√	√	√	4/101/104
7	07	Keyboard d and D	33	√	√	√	4/101/104
8	08	Keyboard e and E	19	√	√	√	4/101/104
9	09	Keyboard f and F	34	√	√	√	4/101/104
10	0A	Keyboard g and G	35	√	√	√	4/101/104
11	0B	Keyboard h and H	36	√	√	√	4/101/104
12	0C	Keyboard i and I	24	√	√	√	4/101/104
13	0D	Keyboard j and J	37	√	√	√	4/101/104
14	0E	Keyboard k and K	38	√	√	√	4/101/104
15	0F	Keyboard l and L	39	√	√	√	4/101/104
16	10	Keyboard m and M[4]	52	√	√	√	4/101/104
17	11	Keyboard n and N	51	√	√	√	4/101/104
18	12	Keyboard o and O[4]	25	√	√	√	4/101/104
19	13	Keyboard p and P[4]	26	√	√	√	4/101/104
20	14	Keyboard q and Q[4]	17	√	√	√	4/101/104

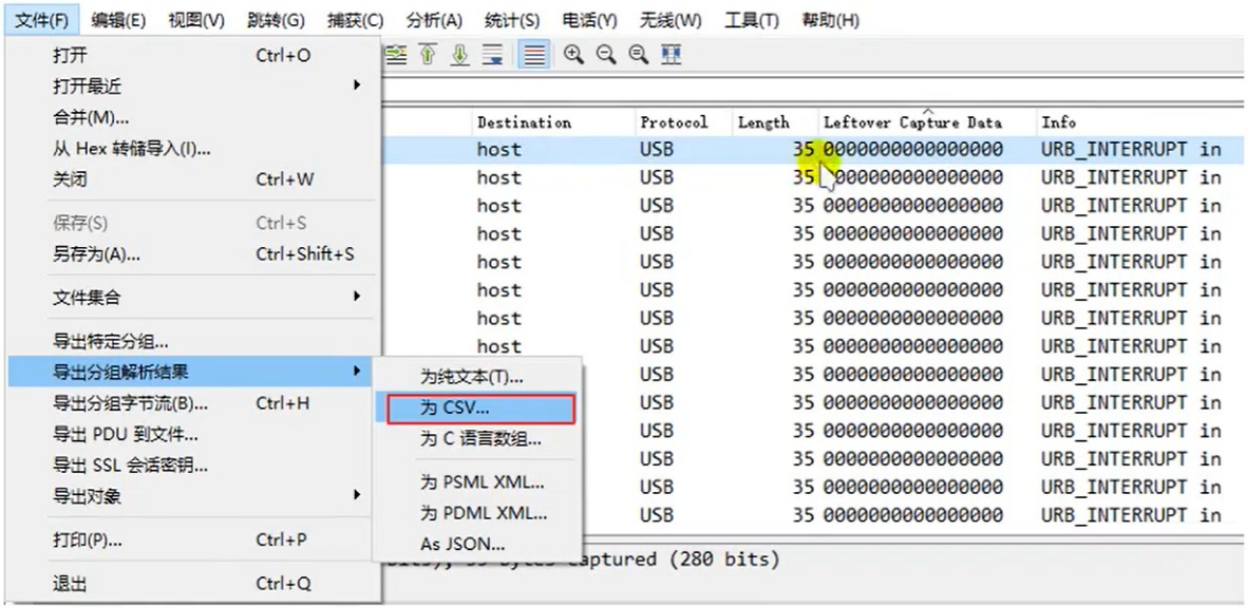
```
# a keyboard maping
mappings = { 0x04:"A",  0x05:"B",  0x06:"C", 0x07:"D", 0x08:"E",
0x09:"F", 0x0A:"G",  0x0B:"H", 0x0C:"I",  0x0D:"J", 0x0E:"K",
0x0F:"L", 0x10:"M", 0x11:"N",0x12:"O",  0x13:"P", 0x14:"Q",
0x15:"R", 0x16:"S", 0x17:"T", 0x18:"U",0x19:"V", 0x1A:"W",
0x1B:"X", 0x1C:"Y", 0x1D:"Z", 0x1E:"1", 0x1F:"2", 0x20:"3",
0x21:"4", 0x22:"5",  0x23:"6", 0x24:"7", 0x25:"8", 0x26:"9",
0x27:"0", 0x28:"\n", 0x2a:"[DEL]",  0X2B:"\t", 0x2C:" ",  0x2D:"-
", 0x2E:"=", 0x2F:"[",  0x30:"]",  0x31:"\\", 0x32:"~", 0x33:";",
0x34:"'", 0x36:",",  0x37:".",0x82:"Casps Lock"}

nums = []
keys = open('usbdata.txt')
data=''
for line in keys:
   line = line.strip()
   if line[0]!='0' or line[1]!='0' 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':
      continue
   strs = str(line.split(":")[2])
   id = "0x"+strs
   if int(id,16) in mappings:
      data +=mappings[int(id,16)]
print data
keys.close()
```

图片上面左边为官方文件，右边为自己写的脚本

USB键盘流量抓取分析

Leftover Capture Data 数据提取方式 1：

文件 → 导出分组解析结果 →为CSV，导出保存成一个.csv文件。

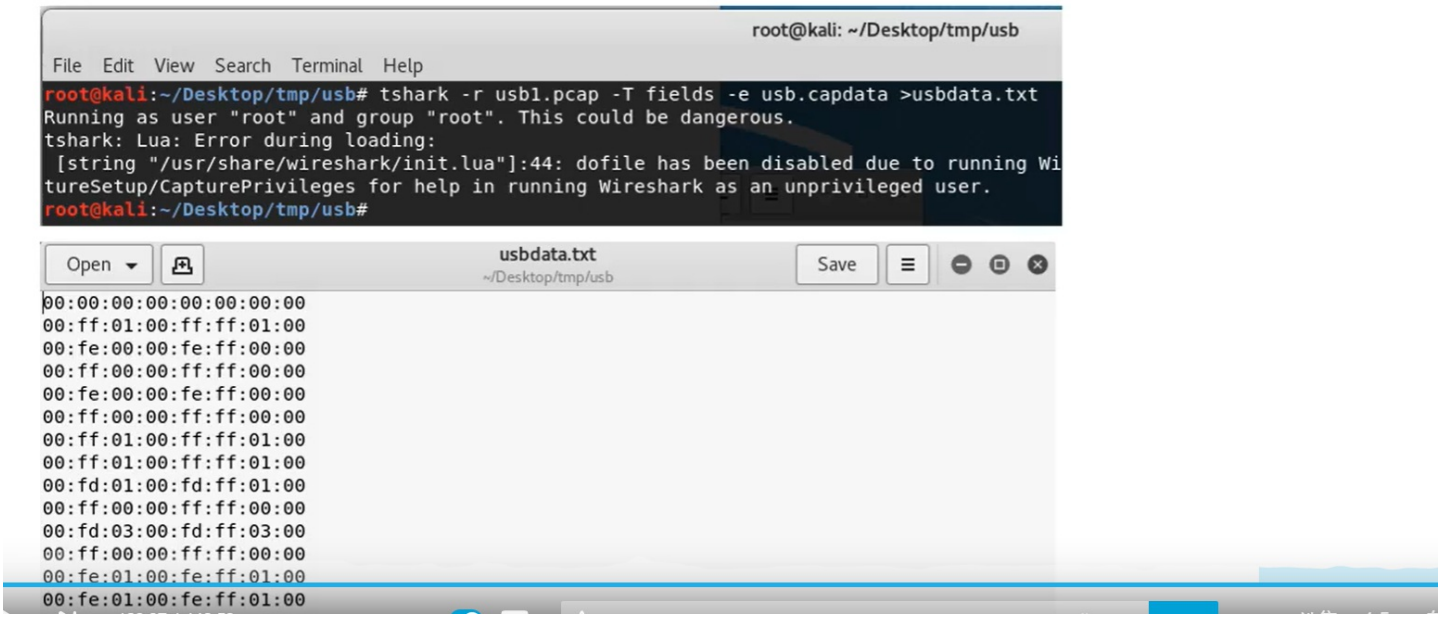


使用wireshark提供的命令行工具tshark

Leftover Capture Data 数据提取方式 2：

使用wireshark提供的命令行工具 tshark，可以将Leftover Capture Data数据单独复制出来。

tshark -r usb1.pcap -T fields -e usb.capdata > usbdata.txt

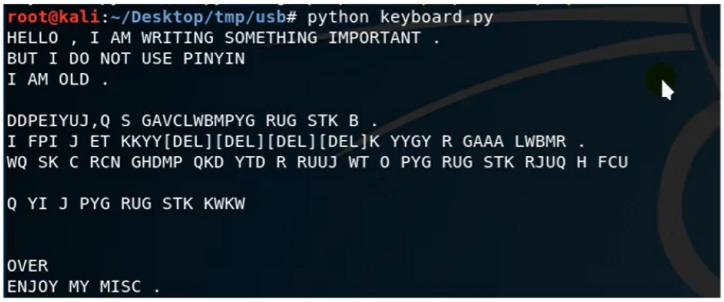


分析 > 导出

USB键盘流量抓取分析

Leftover Capture Data 数据提取方式 2：

python keyborar.py



DDPEIYUJ,Q S GAVCLWBMPYG RUG STK B .
I FPI J ET KKYY[DEL][DEL][DEL][DEL]K YYGY R GAAA LWBMR .
WQ SK C RCN GHDMP QKD YTD R RUUJ WT O PYG RUG STK RJUQ H FCU

Q YI J PYG RUG STK KWKW

用五笔输入法打出

大家注意，我要开始输出福拉格了。
不过是用中文形式输出的。
你可以把下面这句话的拼音作为福拉格提交上去：
我就是福拉格哈哈

USB鼠标流量抓取分析

鼠标流量与键盘流量不同，鼠标移动时表现为连续性，与键盘的离散性不一样。但是实际鼠标产生的数据是离散的。所以同样可以把数据抓取出来，构成二维坐标画出轨迹。

鼠标数据包的数据长度为4个字节，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。

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	Leftover Capture Data	Info
10	1.030704	2.3.1	host	USB	31	00030b00	URB_INTERRUPT in
11	1.030704	2.3.1	host	USB	31	00011100	URB_INTERRUPT in
12	1.051867	2.3.1	host	USB	31	00001800	URB_INTERRUPT in
13	1.052340	2.3.1	host	USB	31	00fd1e00	URB_INTERRUPT in
14	1.052340	2.3.1	host	USB	31	00fb2800	URB_INTERRUPT in
15	1.068194	2.3.1	host	USB	31	00fe2a00	URB_INTERRUPT in
16	1.083866	2.3.1	host	USB	31	00fd2d00	URB_INTERRUPT in
17	1.083866	2.3.1	host	USB	31	00fe2800	URB_INTERRUPT in
18	1.099498	2.3.1	host	USB	31	00002000	URB_INTERRUPT in
19	1.099498	2.3.1	host	USB	31	00000e00	URB_INTERRUPT in
20	1.330480	2.3.1	host	USB	31	00ff0100	URB_INTERRUPT in

第二个字节代表左右偏移；
当值为正时，代表右移多少像素。
当值为负时，代表左移多少像素。

同理，第三个字节代表上下偏移。

对应脚本

USB鼠标流量抓取分析

鼠标流量与键盘流量不同，鼠标移动时表现为连续性，与键盘的离散性不一样。但是实际鼠标产生的数据是离散的。所以同样可以把数据抓取出来，构成二维坐标画出轨迹。

```
nums = []
keys = open('usbdata.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
    print btn_flag
    if btn_flag == 1:
        print posx, posy
keys.close()
```

USB鼠标流量抓取分析

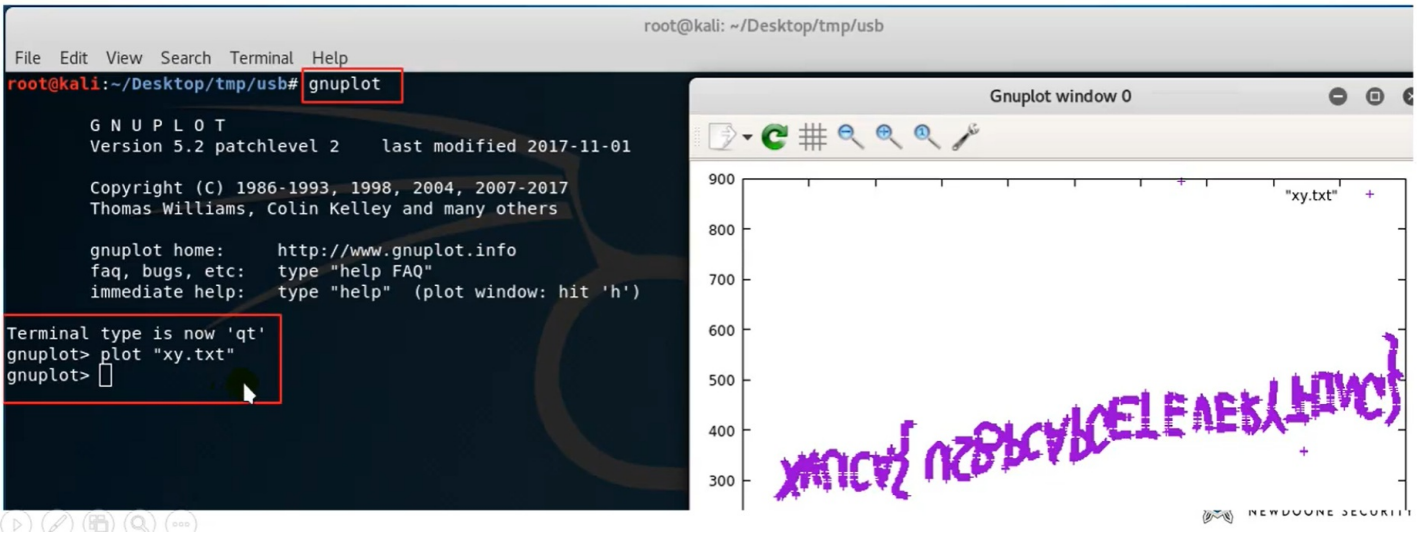
鼠标流量与键盘流量不同，鼠标移动时表现为连续性，与键盘的离散性不一样。但是实际鼠标产生的数据是离散的。所以同样可以把数据抓取出来，构成二维坐标画出轨迹。

```
root@kali:~/Desktop/tmp/usb# python mouse.py
1176 -12
104 268
105 268
108 268
110 269
111 270
113 271
115 273
116 274
118 276
121 279
123 281
```

```
root@kali:~/Desktop/tmp/usb# python mouse.py > xy.txt
root@kali:~/Desktop/tmp/usb# cat xy.txt
1176 -12
104 268
105 268
108 268
110 269
111 270
113 271
115 273
```

把坐标导入到xy.txt

用gnuplot工具把坐标画出来。
gnuplot
plot “xy.txt”



USB鼠标流量抓取分析

Github上有牛人提供了2个工具。

https://github.com/WangYihang/UsbMiceDataHacker

```
Usage :
        python UsbKeyboardHacker.py data.pcap
Tips :
        To use this python script , you must install the tshark first.
        You can use `sudo apt-get install tshark` to install it
```

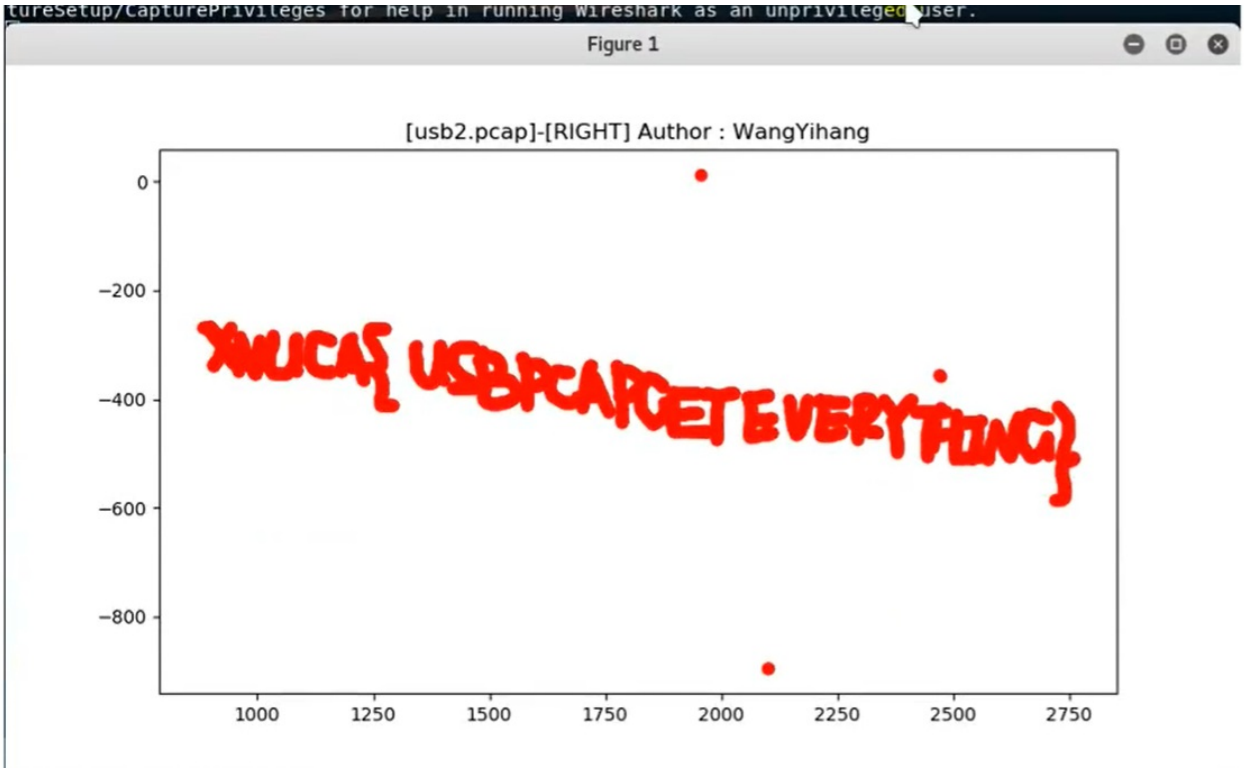
```
root@kali:~/Desktop/tmp/usb# python UsbMiceDataHacker.py -h
Usage :
        python UsbMiceHacker.py data.pcap [LEFT|RIGHT|MOVE|ALL]
Tips :
        To use this python script , you must install the numpy,matplotlib first.
        You can use `sudo pip install matplotlib numpy` to install it
Author :
        WangYihang <wangyihanger@gmail.com>
        If you have any questions , please contact me by email.
        Thank you for using.
```

https://github.com/WangYihang/UsbKeyboardDataHacker

```
Usage :
        python UsbKeyboardHacker.py data.pcap
Tips :
        To use this python script , you must install the tshark first.
        You can use `sudo apt-get install tshark` to install it
```

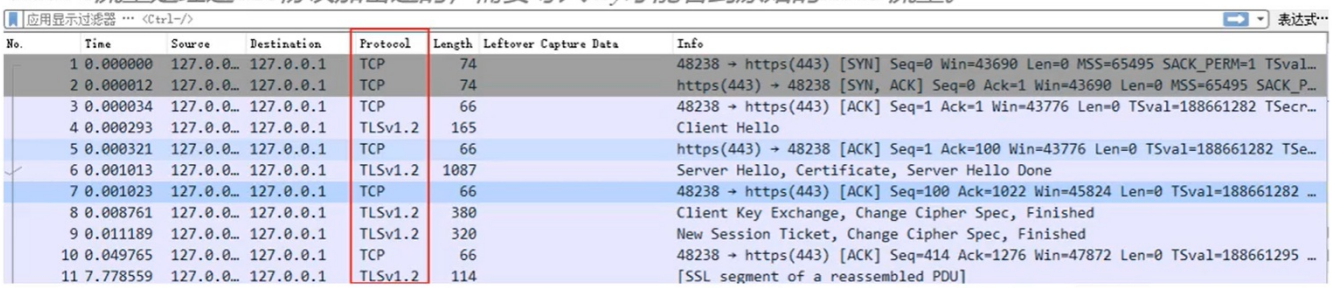
USB鼠标流量抓取分析

```
root@kali:~/Desktop/tmp/usb# python UsbMiceDataHacker.py usb2.pcap RIGHT
tshark -r 'usb2.pcap' -T fields -e usb.capdata > usb.dat
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:44: dofile has been disabled due to running Wireshark as super
```



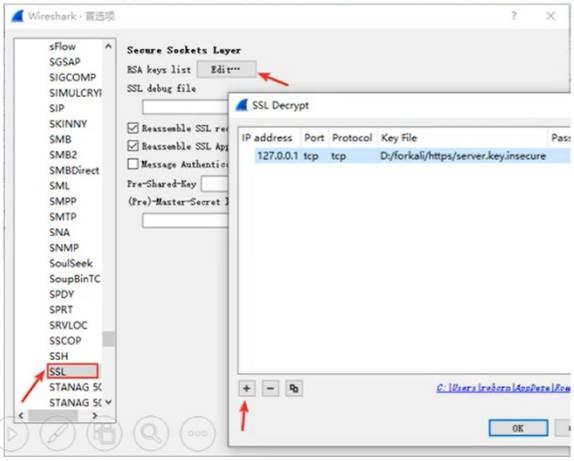
HTTPS流量包文件分析

HTTPS流量是经过TLS协议加密过的，需要导入key才能看到原始的HTTP流量。



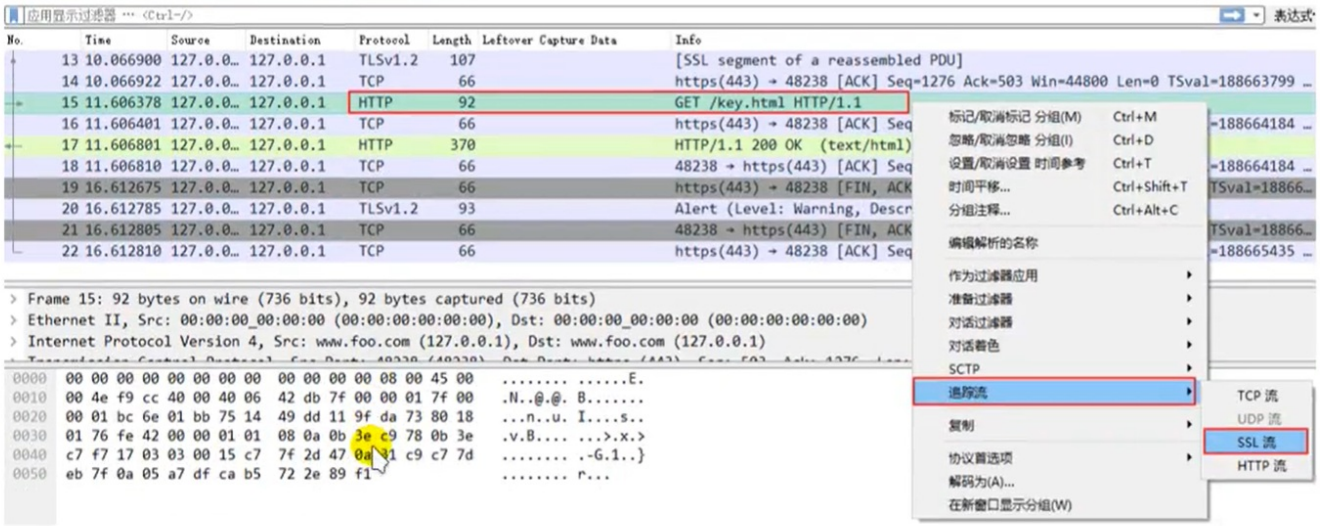
No.	Time	Source	Destination	Protocol	Length	Leftover Capture Data	Info
1	0.000000	127.0.0…	127.0.0.1	TCP	74		48238 → https(443) [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval…
2	0.000012	127.0.0…	127.0.0.1	TCP	74		https(443) → 48238 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_P…
3	0.000034	127.0.0…	127.0.0.1	TCP	66		48238 → https(443) [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=188661282 TSecr…
4	0.000293	127.0.0…	127.0.0.1	TLSv1.2	165		Client Hello
5	0.000321	127.0.0…	127.0.0.1	TCP	66		https(443) → 48238 [ACK] Seq=1 Ack=100 Win=43776 Len=0 TSval=188661282 TSe…
6	0.001013	127.0.0…	127.0.0.1	TLSv1.2	1087		Server Hello, Certificate, Server Hello Done
7	0.001023	127.0.0…	127.0.0.1	TCP	66		48238 → https(443) [ACK] Seq=100 Ack=1022 Win=45824 Len=0 TSval=188661282 …
8	0.008761	127.0.0…	127.0.0.1	TLSv1.2	380		Client Key Exchange, Change Cipher Spec, Finished
9	0.011189	127.0.0…	127.0.0.1	TLSv1.2	320		New Session Ticket, Change Cipher Spec, Finished
10	0.049765	127.0.0…	127.0.0.1	TCP	66		48238 → https(443) [ACK] Seq=414 Ack=1276 Win=47872 Len=0 TSval=188661295 …
11	7.778559	127.0.0…	127.0.0.1	TLSv1.2	114		[SSL segment of a reassembled PDU]

英文版：*Preferences -> Protocols -> SSL -> Edit RSA keys list*
中文版：*编辑 -> 首选项 -> Protocols -> SSL -> Edit RSA keys list*



导入key后出现原始的HTTP流量。

右键→追踪流→SSL流/HTTP流



No.	Time	Source	Destination	Protocol	Length	Leftover Capture Data	Info
13	10.066900	127.0.0…	127.0.0.1	TLSv1.2	107		[SSL segment of a reassembled PDU]
14	10.066922	127.0.0…	127.0.0.1	TCP	66		https(443) → 48238 [ACK] Seq=1276 Ack=503 Win=44800 Len=0 TSval=188663799 …
15	11.606378	127.0.0…	127.0.0.1	HTTP	92		GET /key.html HTTP/1.1
16	11.606401	127.0.0…	127.0.0.1	TCP	66		https(443) → 48238 [ACK] Seq
17	11.606801	127.0.0…	127.0.0.1	HTTP	370		HTTP/1.1 200 OK (text/html)
18	11.606810	127.0.0…	127.0.0.1	TCP	66		48238 → https(443) [ACK] Seq
19	16.612675	127.0.0…	127.0.0.1	TCP	66		https(443) → 48238 [FIN, ACK
20	16.612785	127.0.0…	127.0.0.1	TLSv1.2	93		Alert (Level: Warning, Descr
21	16.612805	127.0.0…	127.0.0.1	TCP	66		48238 → https(443) [FIN, ACK
22	16.612810	127.0.0…	127.0.0.1	TCP	66		https(443) → 48238 [ACK] Seq