USB

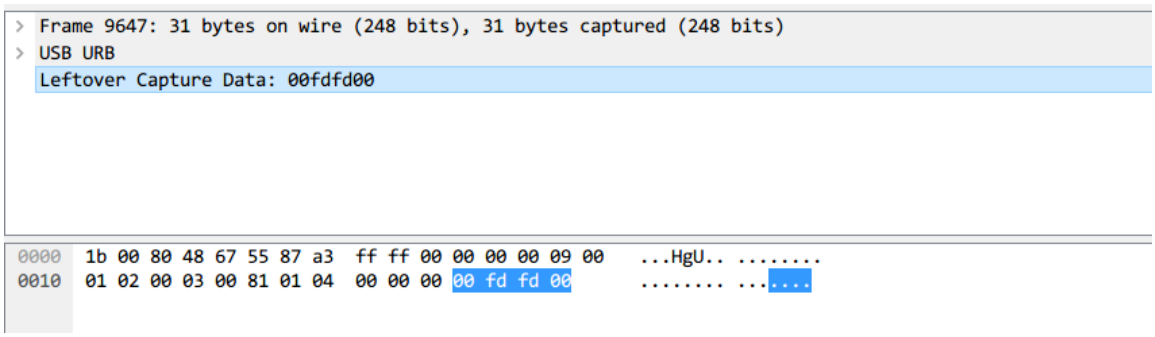
EN | ZH

USB

USB 详述: https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf

- 鼠标协议

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，毕竟计算机表现的连续性信息都是由大量离散信息构成的



每一个数据包的数据区有四个字节，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。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。

得到这些点的信息后, 即可恢复出鼠标移动轨迹

- Tools
- UsbMiceDataHacker
- 键盘协议

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

```
> Frame 307: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)
> USB URB
  Leftover Capture Data: 00fcf300fcfff3ff
```

```
0000  1b 00 f0 8a 42 83 03 97  ff ff 00 00 00 00 09 00   ....B... ........
0010  01 03 00 0a 00 81 01 08  00 00 00 00 fc f3 00 fc   ........ ........
0020  ff f3 ff                                          ...
```

根据 data 值与具体键位的对应关系

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[9]	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

可从数据包恢复出键盘的案件信息

- Tools
- UsbKeyboardDataHacker

参考

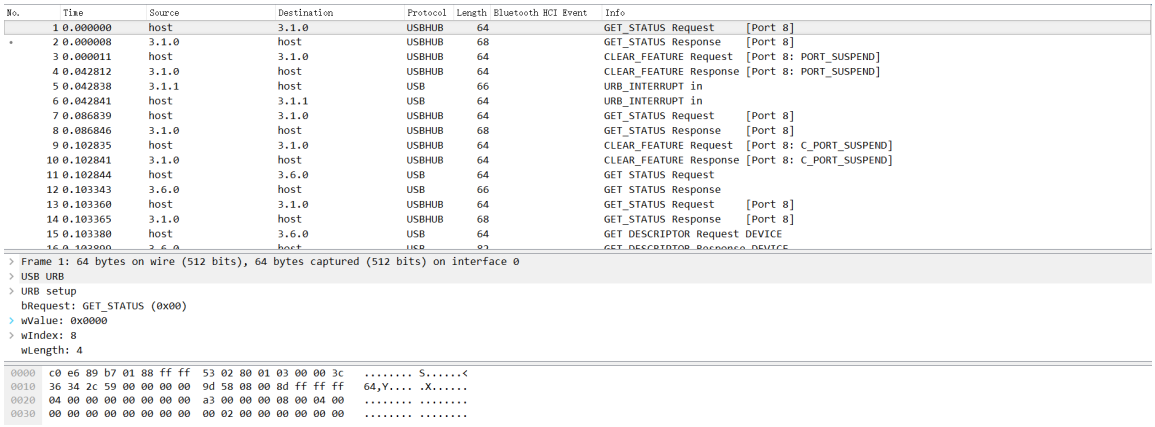
- https://www.anquanke.com/post/id/85218

例题

> `Xman` 三期夏令营排位赛练习题： `AutoKey`

> WP：https://www.cnblogs.com/ECJTUACM-873284962/p/9473808.html

问题描述：



No.	Time	Source	Destination	Protocol	Length	Bluetooth HCI Event	Info
1	0.000000	host	3.1.0	USBHUB	64		GET_STATUS Request [Port 8]
2	0.000008	3.1.0	host	USBHUB	68		GET_STATUS Response [Port 8]
3	0.000011	host	3.1.0	USBHUB	64		CLEAR_FEATURE Request [Port 8: PORT_SUSPEND]
4	0.042812	3.1.0	host	USBHUB	64		CLEAR_FEATURE Response [Port 8: PORT_SUSPEND]
5	0.042838	3.1.1	host	USB	66		URB_INTERRUPT in
6	0.042841	host	3.1.1	USB	64		URB_INTERRUPT in
7	0.086839	host	3.1.0	USBHUB	64		GET_STATUS Request [Port 8]
8	0.086846	3.1.0	host	USBHUB	68		GET_STATUS Response [Port 8]
9	0.102835	host	3.1.0	USBHUB	64		CLEAR_FEATURE Request [Port 8: C_PORT_SUSPEND]
10	0.102841	3.1.0	host	USBHUB	64		CLEAR_FEATURE Response [Port 8: C_PORT_SUSPEND]
11	0.102844	host	3.6.0	USB	64		GET STATUS Request
12	0.103343	3.6.0	host	USB	66		GET STATUS Response
13	0.103360	host	3.1.0	USBHUB	64		GET_STATUS Request [Port 8]
14	0.103365	3.1.0	host	USBHUB	68		GET_STATUS Response [Port 8]
15	0.103380	host	3.6.0	USB	64		GET DESCRIPTOR Request DEVICE

```
> Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
> USB URB
> URB setup
  bRequest: GET_STATUS (0x00)
> wValue: 0x0000
> wIndex: 8
  wLength: 4
```

```
0000  c0 e6 89 b7 01 88 ff ff  53 02 80 01 03 00 00 3c   ........ S......<
0010  36 34 2c 59 00 00 00 00  9d 58 08 00 8d ff ff ff   64,Y.... .X......
0020  04 00 00 00 00 00 00 00  a3 00 00 00 08 00 04 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 02 00 00 00 00 00 00   ........ ........
```

这道题是我参加 `Xman` 三期夏令营选拔赛出的一道题，我们如何对其进行分析？

流量包是如何捕获的?

首先我们从上面的数据包分析可以知道，这是个 `USB` 的流量包，我们可以先尝试分析一下 `USB` 的数据包是如何捕获的。

在开始前，我们先介绍一些 `USB` 的基础知识。 `USB` 有不同的规格，以下是使用 `USB` 的三种方式：

```
l USB UART
l USB HID
l USB Memory
```

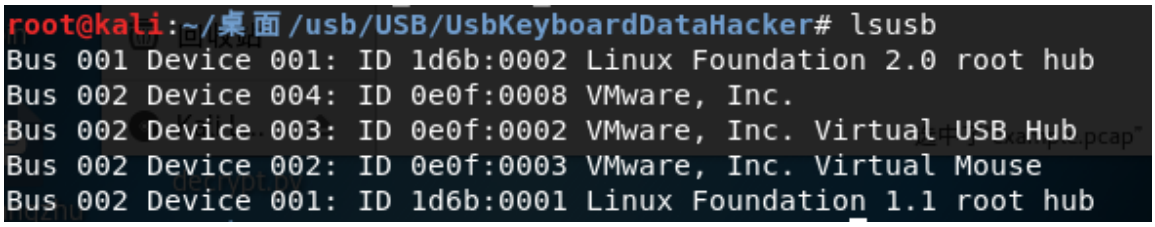
`UART` 或者 `Universal Asynchronous Receiver/Transmitter` 。这种方式下，设备只是简单的将 `USB` 用于接受和发射数据，除此之外就再没有其他通讯功能了。

`HID` 是人性化的接口。这一类通讯适用于交互式，有这种功能的设备有：键盘，鼠标，游戏手柄和数字显示设备。

最后是 `USB Memory` ，或者说是数据存储。 `External HDD` ， `thumb drive/flash drive` 等都是这一类的。

其中使用的最广的不是 `USB HID` 就是 `USB Memory` 了。

每一个 `USB` 设备（尤其是 `HID` 或者 `Memory` ）都有一个供应商 `ID（Vendor ID）` 和产品识别码 `（Product Id）` 。 `Vendor ID` 是用来标记哪个厂商生产了这个 `USB` 设备。 `Product ID` 用来标记不同的产品，他并不是一个特殊的数字，当然最好不同。如下图：



上图是我在虚拟机环境下连接在我电脑上的 `USB` 设备列表，通过 `lsusb` 查看命令。

例如说，我在 `VMware` 下有一个无线鼠标。它是属于 `HID` 设备。这个设备正常的运行，并且通过 `lsusb` 这个命令查看所有 `USB` 设备，现在大家能找出哪一条是这个鼠标吗？？没有错，就是第四个，就是下面这条：

```
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
```

其中，`ID 0e0f:0003` 就是 `Vendor-Product ID` 对，`Vendor ID` 的值是 `0e0f`，并且 `Product ID` 的值是 `0003`。`Bus 002 Device 002` 代表 `usb` 设备正常连接，这点需要记下来。

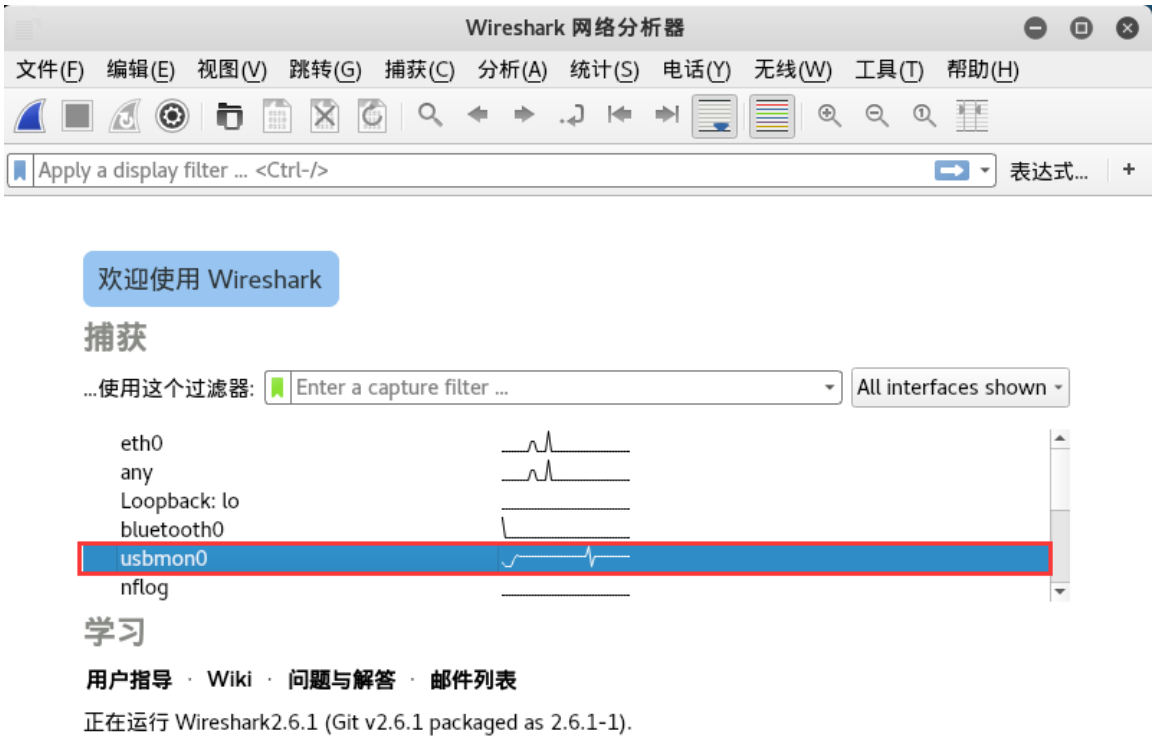
我们用 `root` 权限运行 `Wireshark` 捕获 `USB` 数据流。但是通常来说我们不建议这么做。我们需要给用户足够的权限来获取 `Linux` 中的 `usb` 数据流。我们可以用 `udev` 来达到我们的目的。我们需要创建一个用户组 `usbmon`，然后把我们的账户添加到这个组中。

```
addgroup usbmon
gpasswd -a $USER usbmon
echo 'SUBSYSTEM=="usbmon", GROUP="usbmon", MODE="640"' >
/etc/udev/rules.d/99-usbmon.rules
```

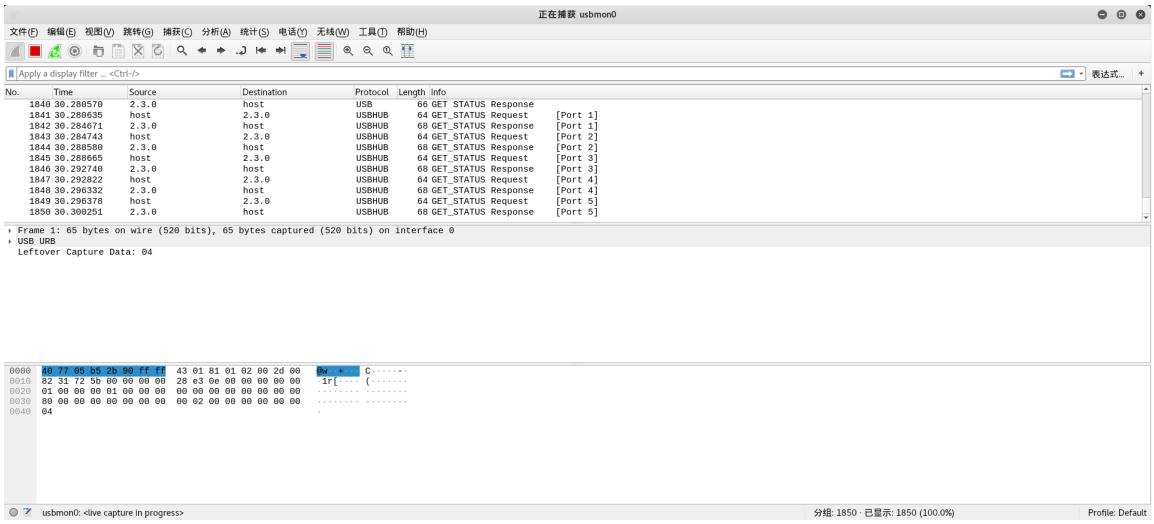
接下来，我们需要 `usbmon` 内核模块。如果该模块没有被加载，我们可以通过以下命令加载该模块：

```
modprobe usbmon
```

打开 `wireshark`，你会看到 `usbmonX` 其中 `X` 代表数字。下图是我们本次的结果（我使用的是 `root`）：



如果接口处于活跃状态或者有数据流经过的时候， `wireshark` 的界面就会把它以波形图的方式显示出来。那么，我们该选那个呢？没有错，就是我刚刚让大家记下来的，这个 X 的数字就是对应这 `USB Bus` 。在本文中是 `usbmon0` 。打开他就可以观察数据包了。



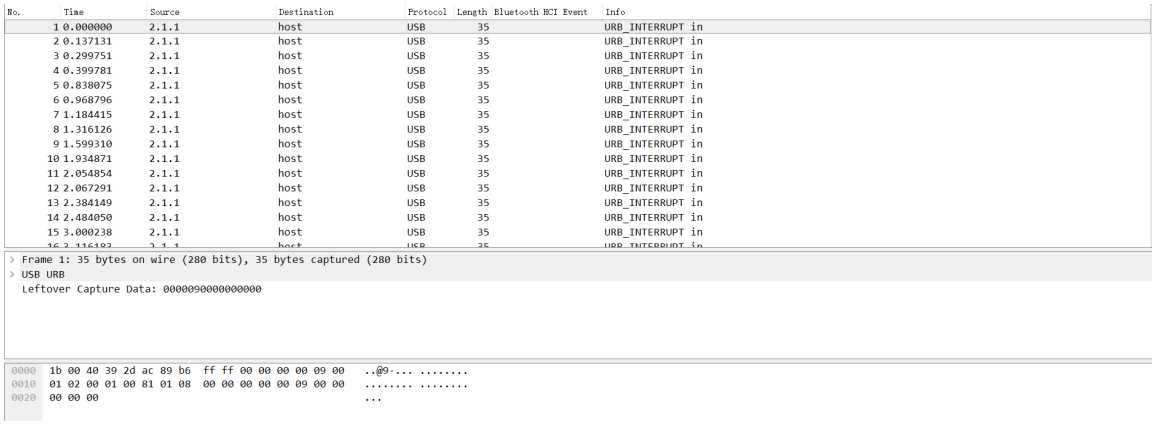
通过这些，我们可以了解到 `usb` 设备与主机之间的通信过程和工作原理，我们可以来对流量包进行分析了。

如何去分析一个 USB 流量包?

根据前面的知识铺垫，我们大致对 `USB` 流量包的抓取有了一个轮廓了，下面我们介绍一下如何分析一个 `USB` 流量包。

`USB` 协议的细节方面参考 `wireshark` 的 `wiki` ： https://wiki.wireshark.org/USB

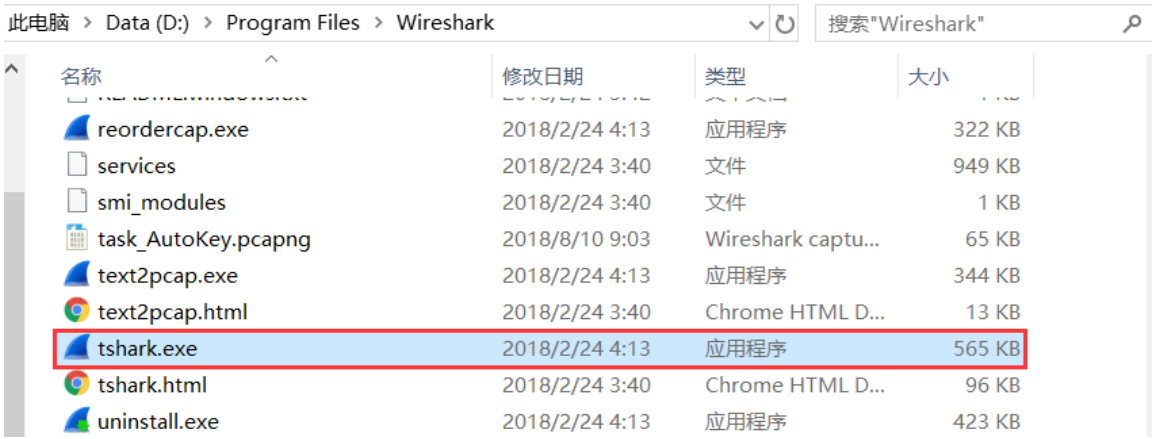
我们先拿 `GitHub` 上一个简单的例子开始讲起：



我们分析可以知道， `USB` 协议的数据部分在 `Leftover Capture Data` 域之中，在 `Mac` 和 `Linux` 下可以用 `tshark` 命令可以将 `leftover capture data` 单独提取出来，命令如下：

```
tshark -r example.pcap -T fields -e usb.capdata //如果想导入usbdata.txt文件中，后面加上参数：>usbdata.txt
```

`Windows` 下装了 `wireshark` 的环境下，在 `wireshark` 目录下有个 `tshark.exe` ，比如我的在 `D:\Program Files\Wireshark\tshark.exe`

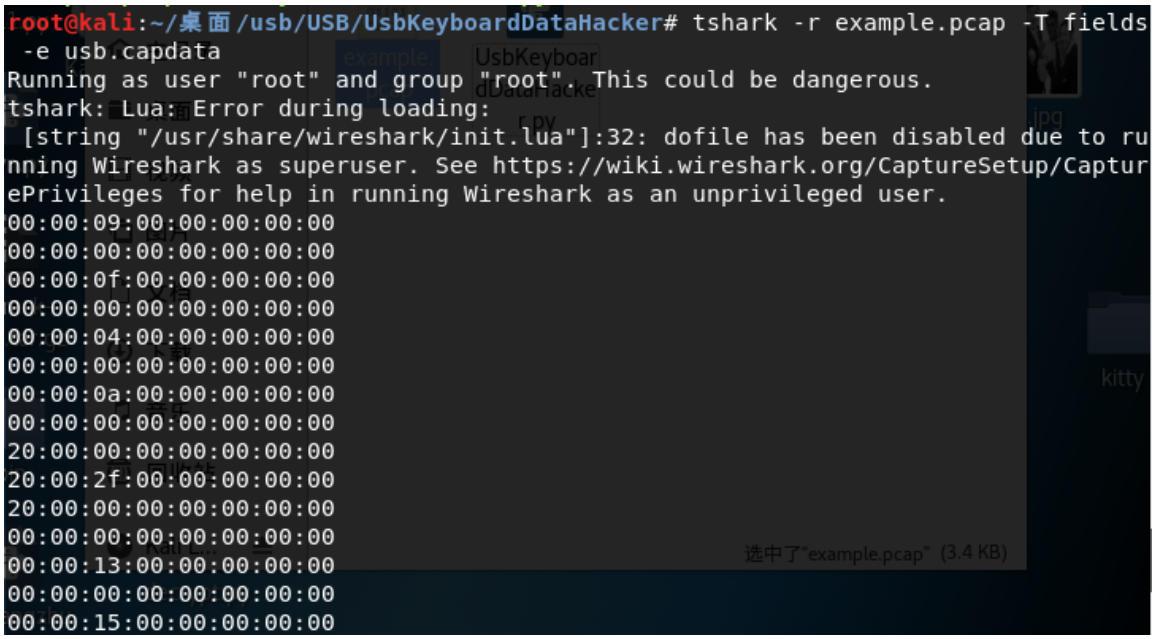


调用 `cmd` ，定位到当前目录下，输入如下命令即可：

```
tshark.exe -r example.pcap -T fields -e usb.capdata //如果想导入usbdata.txt文件中，后面加上参数：>usbdata.txt
```

有关 `tshark` 命令的详细使用参考 `wireshark` 官方文档：
https://www.wireshark.org/docs/man-pages/tshark.html

运行命令并查看 `usbdata.txt` 发现数据包长度为八个字节



关于 `USB` 的特点应用我找了一张图，很清楚的反应了这个问题：

性能	应用	特性
低速(1.5Mbps): ✓交互式设备 ✓10-100kbps	➢键盘，鼠标 ➢手写笔 ➢游戏手柄 ➢虚拟设备 ➢外设	•极低的成本 •易于使用 •热插拔 •同时使用多个外设
全速(12Mbps): ✓电话，音频类 ✓压缩的视频类 ✓500kbps – 10Mbps	➢话音 ➢宽带 ➢音频 ➢麦克风	•较低的成本 •易于使用 •热插拔 •同时使用多个外设 •可保证的带宽 •可保证的延迟
高速(480Mbps): ✓视频，大容量存储 ✓25 – 400Mbps	➢视频 ➢大容量存储 ➢图像 ➢宽带	•低成本 •易于使用 •热插拔 •同时使用多个设备 •可保证的带宽 •可保证的延迟 •高带宽

这里我们只关注 `USB` 流量中的键盘流量和鼠标流量。

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` 。

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。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。

我翻阅了大量的 `USB` 协议的文档，在这里我们可以找到这个值与具体键位的对应关系：
https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf

`usb keyboard` 的映射表 根据这个映射表将第三个字节取出来，对应对照表得到解码：

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[9]	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

我们写出如下脚本：

```
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:"    ", 0x2C:" ",
0x2D:"-", 0x2E:"=", 0x2F:"[",  0x30:"]",  0x31:"\\", 0x32:"~", 0x33:";",
0x34:"'", 0x36:",",  0x37:"." }
nums = []
keys = open('usbdata.txt')
for line in keys:
    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
    nums.append(int(line[6:8],16))
    # 00:00:xx:....
keys.close()
output = ""
for n in nums:
    if n == 0 :
        continue
```

```
    if n in mappings:
        output += mappings[n]
    else:
        output += '[unknown]'
print('output :n' + output)
```

结果如下:

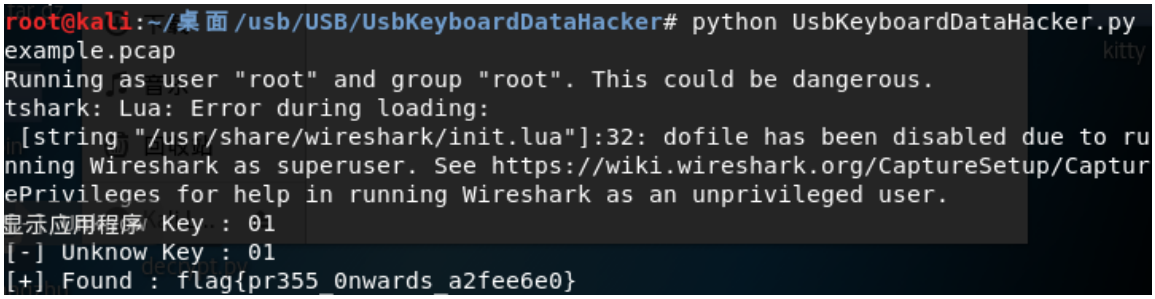
```
root@kali:~/桌面/usb/USB/UsbKeyboardDataHacker# python pwn.py
output :nFLAGPR3550NWARDSA2FEE6E0
```

我们把前面的整合成脚本，得:

```
#!/usr/bin/env python

import sys
import os

DataFileName = "usb.dat"

presses = []

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>"}

def main():
    # check argv
    if len(sys.argv) != 2:
        print "Usage : "
        print "        python UsbKeyboardHacker.py data.pcap"
        print "Tips : "
        print "        To use this python script , you must install the
tshark first."
        print "        You can use `sudo apt-get install tshark` to install
it"
```

```
        print "        Thank you for using."
        exit(1)

    # get argv
    pcapFilePath = sys.argv[1]

    # get data of pcap
    os.system("tshark -r %s -T fields -e usb.capdata > %s" % (pcapFilePath,
DataFileName))

    # read data
    with open(DataFileName, "r") as f:
        for line in f:
            presses.append(line[0:-1])
    # handle
    result = ""
    for press in presses:
        Bytes = press.split(":")
        if Bytes[0] == "00":
            if Bytes[2] != "00":
                result += normalKeys[Bytes[2]]
        elif Bytes[0] == "20": # shift key is pressed.
            if Bytes[2] != "00":
                result += shiftKeys[Bytes[2]]
        else:
            print "[-] Unknow Key : %s" % (Bytes[0])
    print "[+] Found : %s" % (result)

    # clean the temp data
    os.system("rm ./%s" % (DataFileName))


if __name__ == "__main__":
    main()
```

效果如下：



另外贴上一份鼠标流量数据包转换脚本：

```
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
if btn_flag == 1 :
    print posx , posy
keys.close()
```

键盘流量数据包转换脚本如下：

```
nums=
[0x66,0x30,0x39,0x65,0x35,0x34,0x63,0x31,0x62,0x61,0x64,0x32,0x78,0x33,0x38,0x

s=''
for x in nums:
    s+=chr(x)
print s
mappings = { 0x41:"A",  0x42:"B",  0x43:"C", 0x44:"D", 0x45:"E", 0x46:"F",
0x47:"G",  0x48:"H", 0x49:"I",  0x4a:"J", 0x4b:"K", 0x4c:"L", 0x4d:"M",
0x4e:"N",0x4f:"O",  0x50:"P", 0x51:"Q", 0x52:"R", 0x53:"S", 0x54:"T",
0x55:"U",0x56:"V", 0x57:"W", 0x58:"X", 0x59:"Y", 0x5a:"Z", 0x60:"0",
0x61:"1", 0x62:"2", 0x63:"3", 0x64:"4",  0x65:"5", 0x66:"6", 0x67:"7",
0x68:"8", 0x69:"9", 0x6a:"*", 0x6b:"+",  0X6c:"separator", 0x6d:"-",
0x6e:".", 0x6f:"/" }
output = ""
for n in nums:
    if n == 0 :
        continue
    if n in mappings:
        output += mappings[n]
    else:
        output += '[unknown]'
print 'output :\n' + output
```

那么对于 `xman` 三期夏令营排位赛的这道题，我们可以模仿尝试如上这个例子：

首先我们通过 `tshark` 将 `usb.capdata` 全部导出：

```
tshark -r task_AutoKey.pcapng -T fields -e usb.capdata //如果想导入
usbdata.txt文件中，后面加上参数：>usbdata.txt
```

结果如下：

```
root@kali:~/桌面/usb/USB# tshark -r task_AutoKey.pcapng -T fields -e usb.capdata
 >usbdata.txt
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:32: dofile has been disabled due to ru
nning Wireshark as superuser. See https://wiki.wireshark.org/CaptureSetup/Captur
ePrivileges for help in running Wireshark as an unprivileged user.
root@kali:~/桌面/usb/USB# ls
task_AutoKey.pcapng  usbdata.txt          UsbKeyboardDataHacker.py
usb.dat              UsbKeyboardDataHacker
```

我们用上面的 `python` 脚本将第三个字节取出来，对应对照表得到解码：

```
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:"    ", 0x2C:" ",
0x2D:"-", 0x2E:"=", 0x2F:"[",  0x30:"]",  0x31:"\\", 0x32:"~", 0x33:";",
0x34:"'", 0x36:",",  0x37:"." }
nums = []
keys = open('usbdata.txt')
for line in keys:
    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
    nums.append(int(line[6:8],16))
    # 00:00:xx:....
keys.close()
output = ""
for n in nums:
    if n == 0 :
        continue
    if n in mappings:
        output += mappings[n]
    else:
        output += '[unknown]'
print('output :n' + output)
```

运行结果如下：

```
root@kali:~/桌面/usb/USB# python pwn.py
output :n[unknown]A[unknown]UTOKEY''.DECIPHER'[unknown]MPLRVFFCZEYOUJFJKYBXGZVDG
QAURKXZOLKOLVTUFBLRNJESQITWAHXNSIJXPNMPLSHCJBTYHZEALOGVIAAISSPLFHLFSWFEHJNCRWHTI
NSMAMBVEXO[DEL]PZE[DEL]IZ'
```

```
output
:n[unknown]A[unknown]UTOKEY''.DECIPHER'[unknown]MPLRVFFCZEYOUJFJKYBXGZVDGQAURK
```

我们可以看出这是自动密匙解码，现在的问题是在我们不知道密钥的情况下应该如何解码呢？

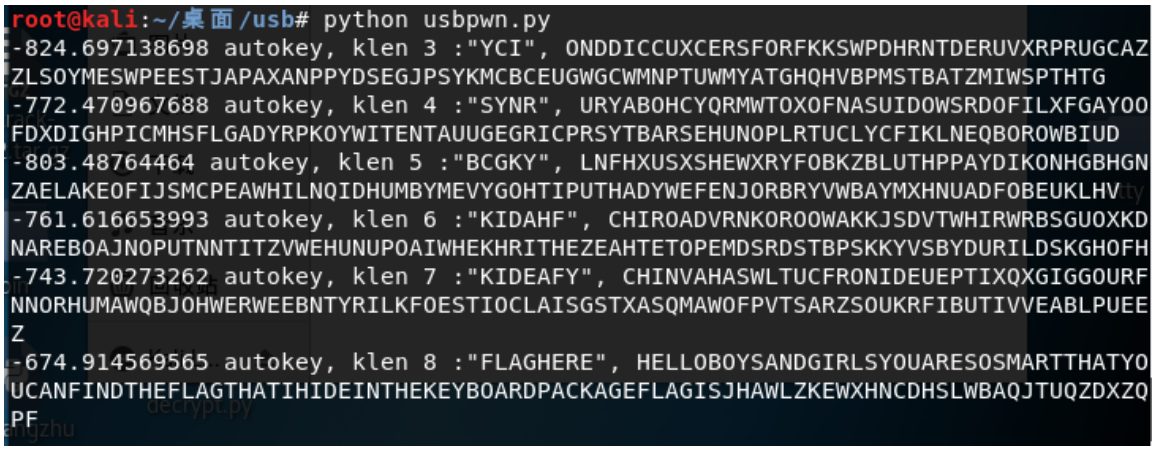
我找到了如下这篇关于如何爆破密匙：
http://www.practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-autokey-cipher/

爆破脚本如下：

```
from ngram_score import ngram_score
from pycipher import Autokey
import re
from itertools import permutations

qgram = ngram_score('quadgrams.txt')
trigram = ngram_score('trigrams.txt')

ctext =
'MPLRVFFCZEYOUJFJKYBXGZVDGQAURKXZOLKOLVTUFBLRNJESQITWAHXNSIJXPNMPLSHCJBTYHZEAL


ctext = re.sub(r'[^A-Z]','',ctext.upper())
# keep a list of the N best things we have seen, discard anything else

class nbest(object):
    def __init__(self,N=1000):
        self.store = []
        self.N = N

    def add(self,item):
        self.store.append(item)
        self.store.sort(reverse=True)
        self.store = self.store[:self.N]

    def __getitem__(self,k):
        return self.store[k]

    def __len__(self):
        return len(self.store)

#init
N=100
for KLEN in range(3,20):
    rec = nbest(N)
    for i in permutations('ABCDEFGHIJKLMNOPQRSTUVWXYZ',3):
        key = ''.join(i) + 'A'*(KLEN-len(i))
        pt = Autokey(key).decipher(ctext)
        score = 0
        for j in range(0,len(ctext),KLEN):
            score += trigram.score(pt[j:j+3])
        rec.add((score,''.join(i),pt[:30]))

    next_rec = nbest(N)
    for i in range(0,KLEN-3):
        for k in xrange(N):
            for c in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
                key = rec[k][1] + c
                fullkey = key + 'A'*(KLEN-len(key))
                pt = Autokey(fullkey).decipher(ctext)
```

```
                score = 0
                for j in range(0,len(ctext),KLEN):
                    score += qgram.score(pt[j:j+len(key)])
                next_rec.add((score,key,pt[:30]))
        rec = next_rec
        next_rec = nbest(N)
    bestkey = rec[0][1]
    pt = Autokey(bestkey).decipher(ctext)
    bestscore = qgram.score(pt)
    for i in range(N):
        pt = Autokey(rec[i][1]).decipher(ctext)
        score = qgram.score(pt)
        if score > bestscore:
            bestkey = rec[i][1]
            bestscore = score
    print bestscore,'autokey,
klen',KLEN,':"'+bestkey+'",',Autokey(bestkey).decipher(ctext)
```

跑出来的结果如下：



我们看到了 `flag` 的字样，整理可得如下：

```
-674.914569565 autokey, klen 8 :"FLAGHERE",
HELLOBOYSANDGIRLSYOUARESOSMARTTHATYOUCANFINDTHEFLAGTHATIHIDEINTHEKEYBOARDPACKA
```

我们把字段进行分割看：

```
HELLO
BOYS
AND
GIRLS
YOU
ARE
SO
SMART
THAT
YOU
CAN
FIND
```

```
THE
FLAG
THAT
IH
IDE
IN
THE
KEY
BOARD
PACKAGE
FLAG
IS
JHAWLZKEWXHNCDHSLWBAQJTUQZDXZQPF
```

最后的 `flag` 就是 `flag{JHAWLZKEWXHNCDHSLWBAQJTUQZDXZQPF}`

参考文献

- https://www.cnblogs.com/ECJTUACM-873284962/p/9473808.html
- https://blog.csdn.net/songze_lee/article/details/77658094
- https://wiki.wireshark.org/USB
- https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf
- https://www.wireshark.org/docs/man-pages/tshark.html
- http://www.practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-autokey-cipher/
- https://hackfun.org/2017/02/22/CTF%E4%B8%AD%E9%82%A3%E4%BA%9B%E8%84%91%E6%B4%9E%E5%A4%A7%E5%BC%80%E7%9A%84%E7%BC%96%E7%A0%81%E5%92%8C%E5%8A%A0%E5%AF%86/

评论