PKU GeekGame 1st writeup

RannRu

2021.11.20

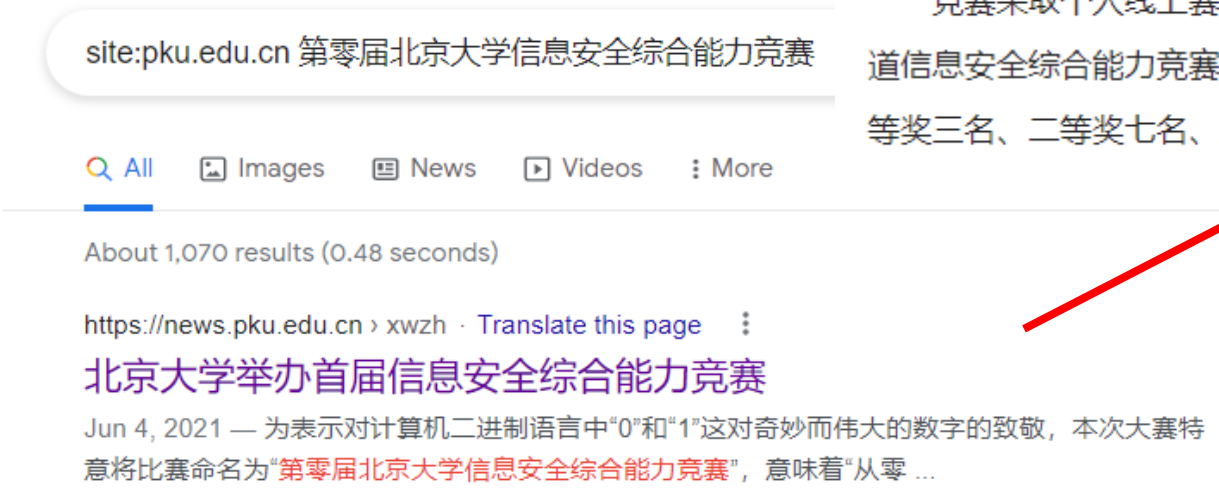
签到

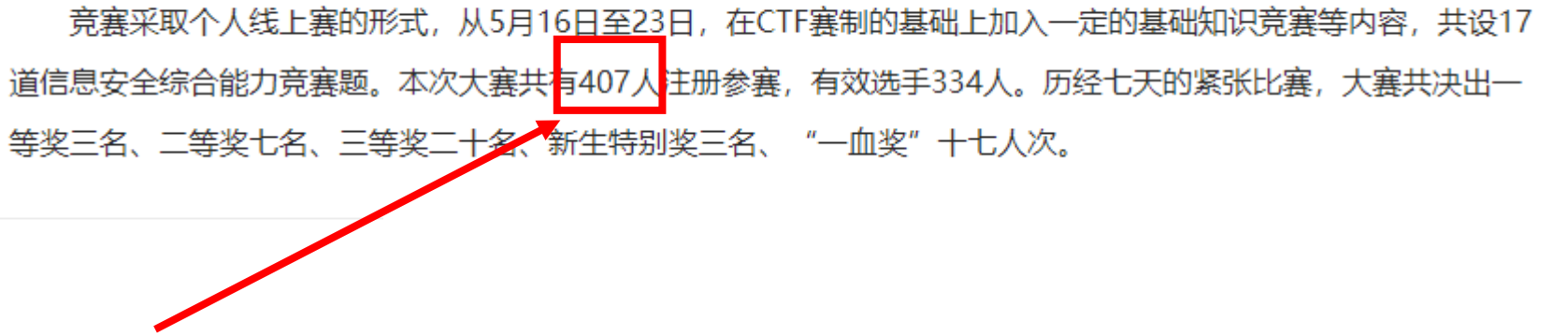
- 一开始用**Acrobat**打开pdf文件，是一堆奇怪的字符，盲猜使用一些数学符号用等特殊字体来显示后包装到pdf内
- 复制出来完全看不懂但我不相信还有蹊跷
- 遂用**Chrome**打开
- fa{aeAGetTm@ekaev!
- lgHv__ra_ieGeGm_1}
- 注意到交替读两行的字符可以得到flag
- 于是结果为
- flag{Have_A_Great_Time@GeekGame_v1!}

小北问答 Remake

- ~~我最擅长搜索了（bushi~~
- #1：北京大学燕园校区有理科 1 号楼到理科 X 号楼，但没有理科 (X+1) 号及之后的楼。X 是？
- 校内同学熟知有理科五号楼，遂X=5

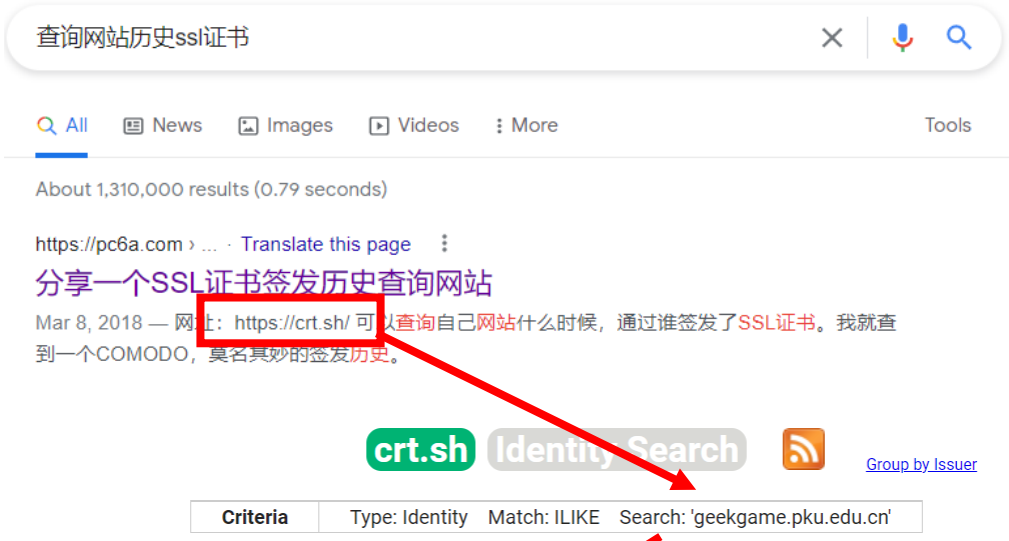
- #2：上一届（第零届）比赛的总注册人数有多少？
- 正确使用Google，409

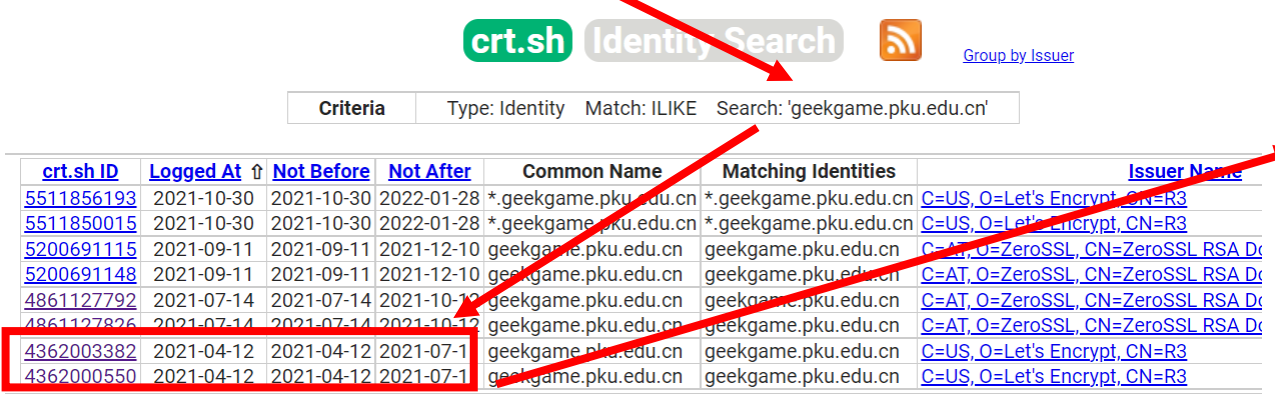


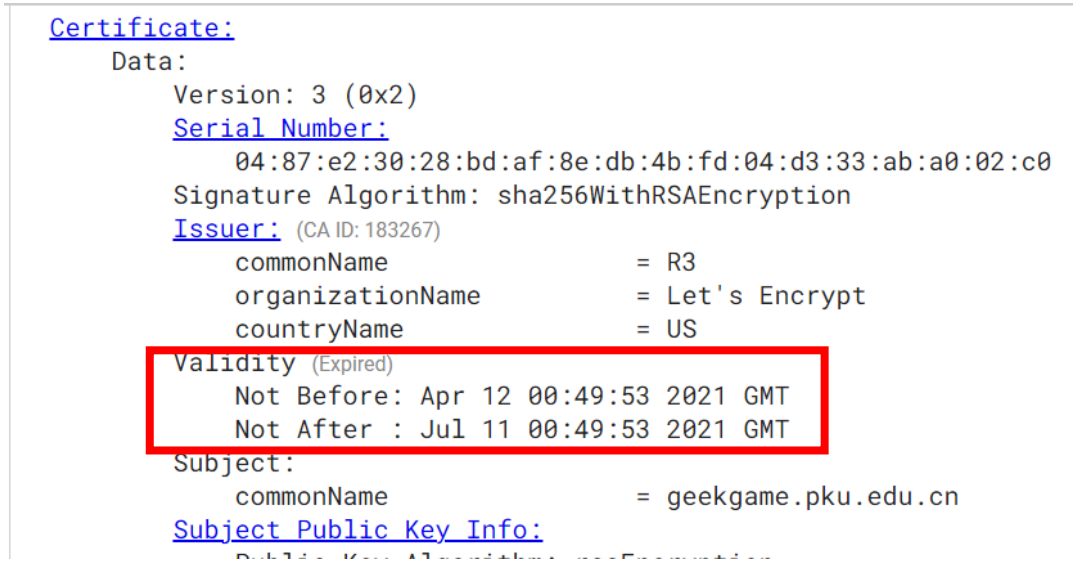


小北问答 Remake

- #3：geekgame.pku.edu.cn 的 HTTPS 证书曾有一次忘记续期了，发生过期的时间是？
- 虽然是在Google用中文查的，但好像也挺有效，秒数个位刚好和格式一致，只需要换下时区即可，2021-07-11T08:49:53+08:00







小北问答 Remake

- #4：2020 年 DEFCON CTF 资格赛签到题的 flag 是？
- Google找到比赛网站https://archive.ooo/c/welcome-to-dc2020-quals/358/
- 下载压缩包即可获得flag OOO{this_is_the_welcome_flag}

- #5：在大小为 672328094 * 386900246 的方形棋盘上放 3 枚（相同的）皇后且它们互不攻击，有几种方法？
- 有点麻烦，其实写了个爆搜打了张二维表，想去OEIS（经典查数列网站）上查查看有没有矩阵，发现查不到。试了试对角线发现了n*n的是有的 https://oeis.org/A047659，一般来说还会有计算方法和参考文献，翻了翻发现了这个，刚好就是需要的，拉进Python一通大整数计算即可，结果是

  293352326016613792399840 93

  096470574938828065255775 36

In general, for m <= n, n >= 3, the number of ways to place 3 nonattacking queens on an m X n board is n^3/6*(m^3 - 3*m^2 + 2*m) - n^2/2*(3*m^3 - 9*m^2 + 6*m) + n/6*(2*m^4 + 20*m^3 - 77*m^2 + 58*m) - 1/24*(39*m^4 - 82*m^3 - 36*m^2 + 88*m) + 1/16*(2*m - 4*n + 1)*(1 + (-1)^(m+1)) + 1/2*(1 + abs(n - 2*m + 3) - abs(n - 2*m + 4))*(1/24*((n - 2*m + 11)^4 - 42*(n - 2*m + 11)^3 + 656*(n - 2*m + 11)^2 - 4518*(n - 2*m + 11) + 11583) - 1/16*(4*m - 2*n - 1)*(1 + (-1)^(n+1))) [Panos Louridas, idee & form 93/2007, pp. 2936-2938]. - Vaclav Kotesovec, Feb 20 2016

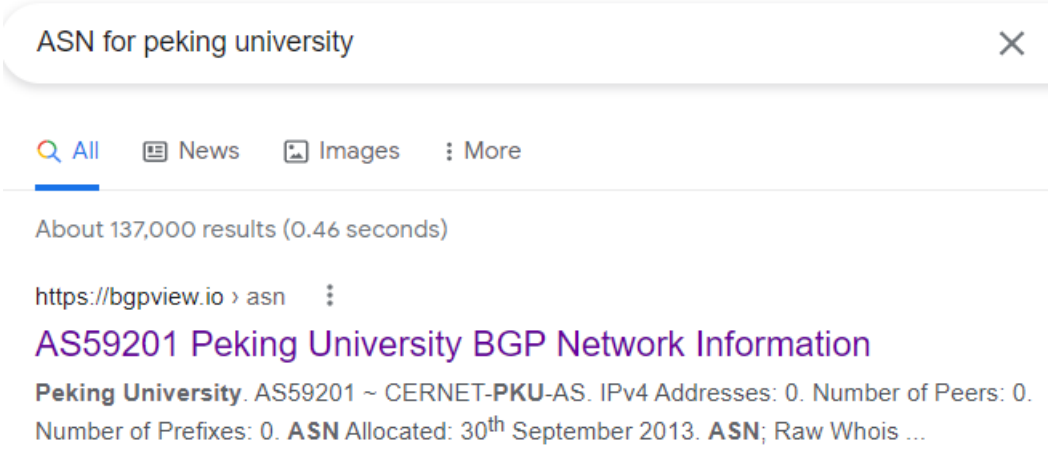
小北问答 Remake

- #6：上一届（第零届）比赛的“小北问答1202”题目会把所有选手提交的答案存到 SQLite 数据库的一个表中，这个表名叫？
- 在比赛主页可以找到历史Github仓库，里面打开小北问答中的源码 https://github.com/PKU-GeekGame/geekgame-0th/blob/main/src/choice/game/db.py

```
def init_db():
    db = get_db()
    db.executescript('''
        create table if not exists submits (
            uid int,
            submit_ts int,
            answers_json text
        );
    ''')
```

- 于是答案为 submits

- #7：国际互联网由许多个自治系统（AS）组成。北京大学有一个自己的自治系统，它的编号是？
- Google，永远滴神
- 答案是 AS59201

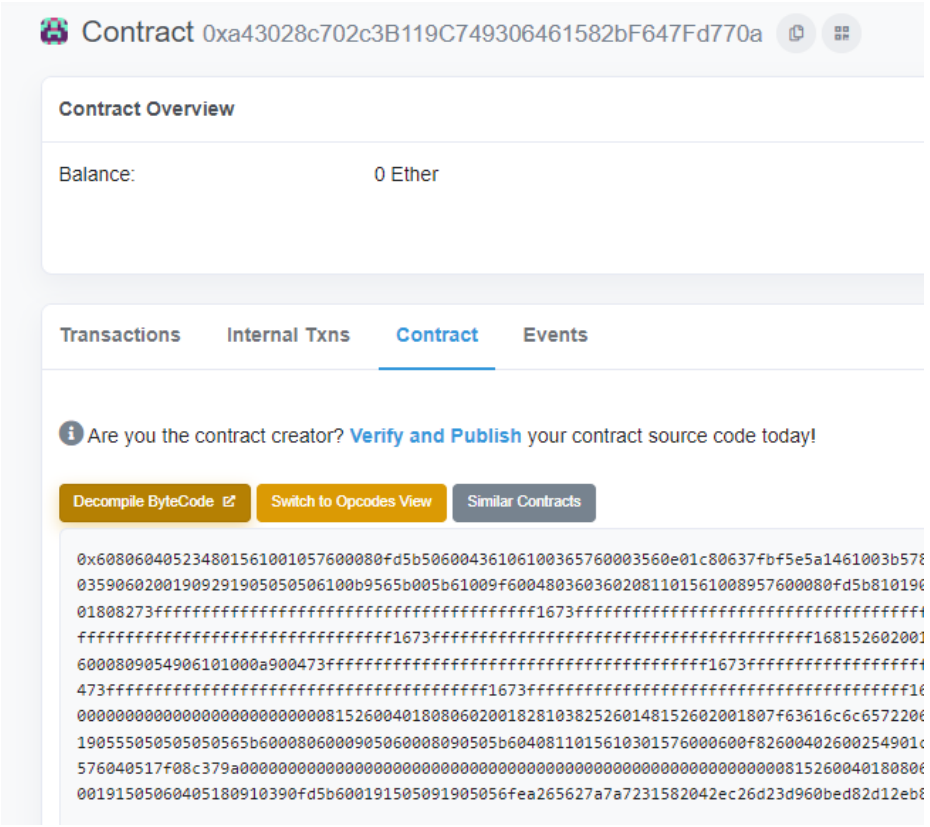


小北问答 Remake

- #8：截止到 2021 年 6 月 1 日，完全由北京大学信息科学技术学院下属的中文名称最长的实验室叫？
- 这题我不好说，大概是瞎找的，打开信科官网https://eecs.pku.edu.cn/
- 第一个是就是电子学系，点开第一个实验室怎么名字就巨长，然后随便翻翻其他网站都没比他长的，于是 就当作是答案了（反正可以重复提交）
- 没想到一次就对了 区域光纤通信网与新型光通信系统国家重点实验室

共享的机器

- 学习了下智能合约是什么玩意，简单来说大概就是一份代码，可以通过交易来付钱运行公用函数
- 打开给的网站，点到Contract上（诶怎么别人的都是源代码你这是ByteCode）
- 原来这个没验证，需要逆向一下，一看它自带了个decompile
- 点开发现一个奇怪的函数，读一下逻辑
- 大概就是输入经过一通涉及到storage[2]的可逆
- 运算后和storage[3]比较，如果不相同就不是flag
- 那么只要搞清楚怎么读storage就好了




```
def unknownded0677d(uint256 _param1) payable:
  require calldata.size - 4 >= 32
  idx = 0
  s = 0
  while idx < 64:
      idx = idx + 1
      s = s or (Mask(256, -4 * idx, _param1) >> 4 * idx) + (5 * idx) + (7 * Mask(256, -4 * idx, stor2) >> 4 * idx) % 16 << 4 * idx
      continue
  if stor3 != 0:
      revert with 0, 'this is not the real flag!'
  return 1
```

共享的机器

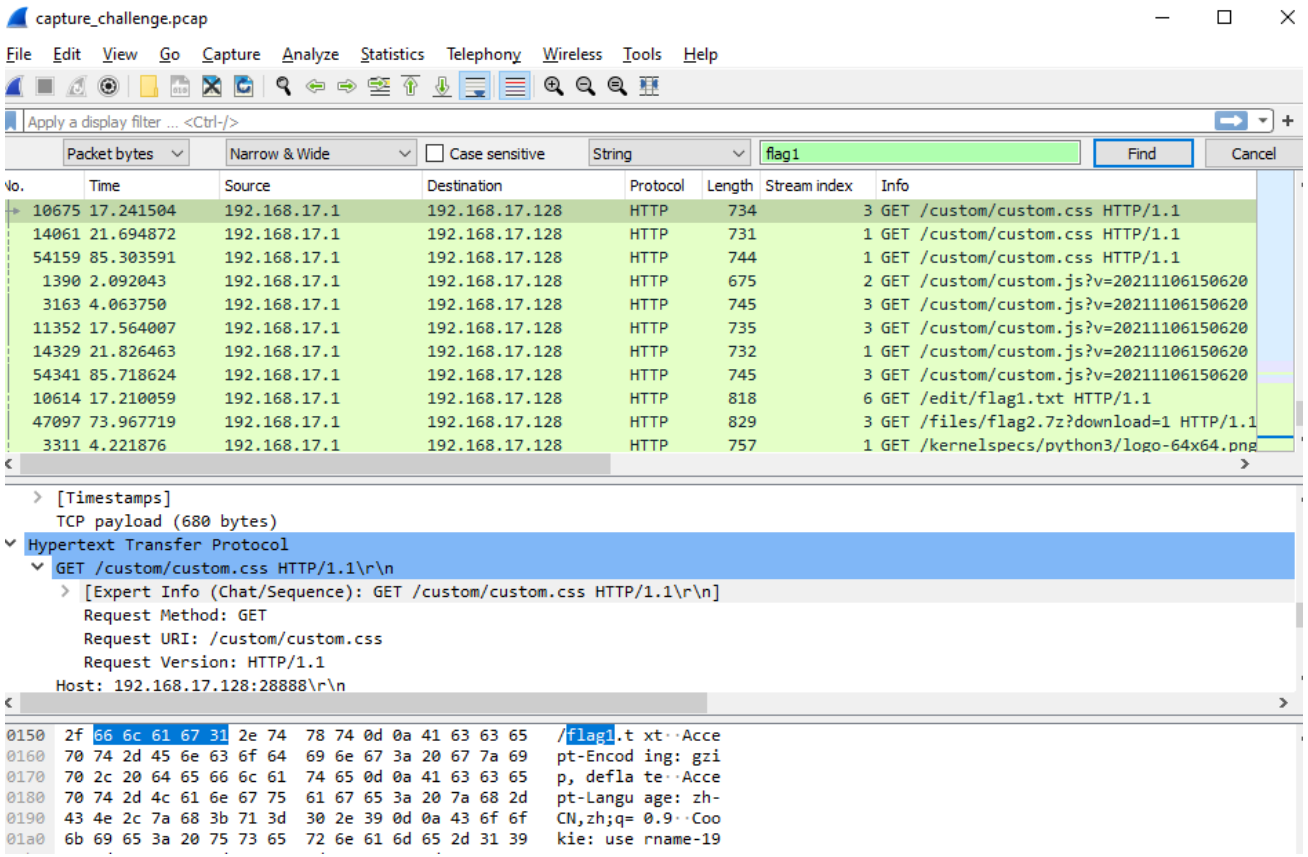
- 说起来那个Mask有点看不懂，找了个可读一点的网站https://www.contract-library.com/contracts/Ethereum/A43028C702C3B119C749306461582BF647FD770A?line=null&tab=Decompiled `v0 = v0 | ((varg0 >> (v2 << 2)) + 5 * v2 + 7 * (stor_2 >> (v2 << 2)) & 0xf) << (v2 << 2);`
- 关于storage怎么读，上网搜了搜发现有趣的nodejs API
- https://web3js.readthedocs.io/en/v1.2.11/web3-eth-contract.html
- 像这样读出来storage之后，随便写个python算一算即可

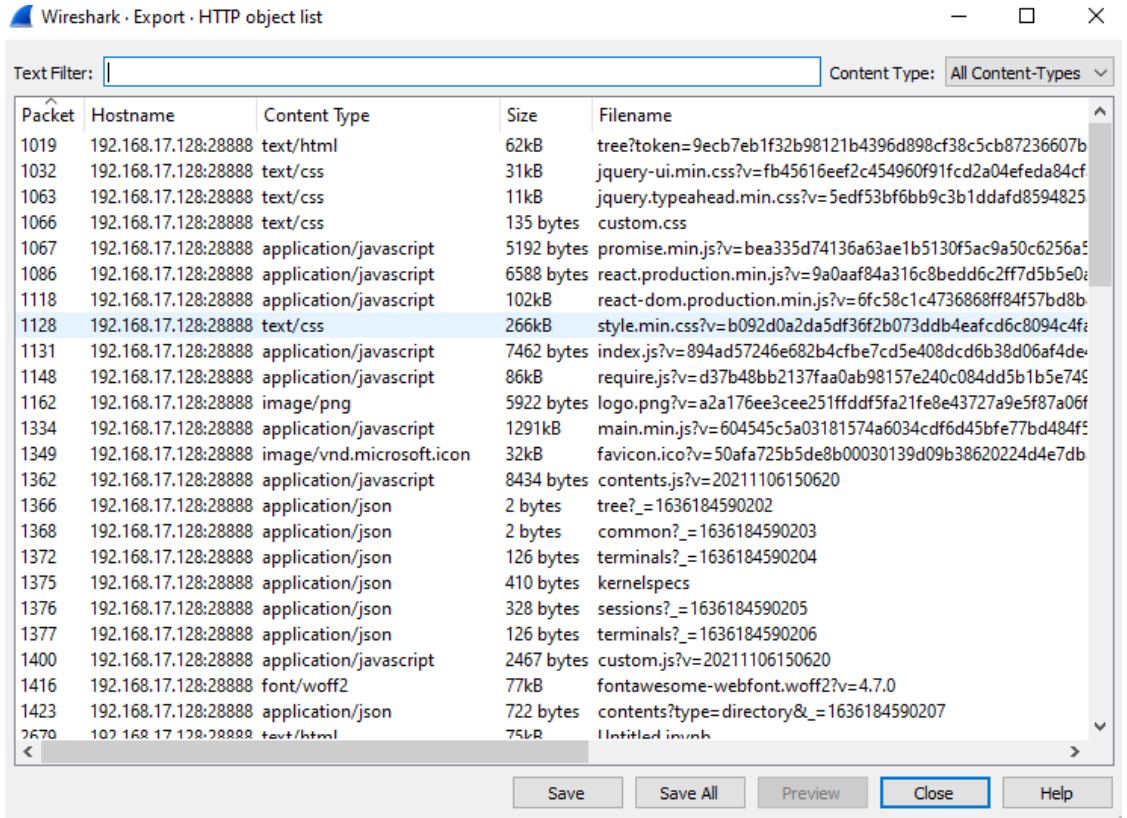
```
stor2 = "0x15eea4b2551f0c96d02a5d62f84cac8112690d68c47b16814e221b8a37d6c4d3"
stor3 = "0x293edea661635aabcd6deba615ab813a7610c1cfb9efb31ccc5224c0e4b37372"

s2 = int(stor2, 16)
s3 = int(stor3, 16)

ans = 0

for i in range(64):
    j = i << 2
    tmp = (s3 >> j) & 0xf
    tmp2 = 5 * i + 7 * (s2 >> j)
    ans |= ((tmp-tmp2)& 0xf) << j

print(bytes.fromhex(hex(ans)[2:]))

b'flag{N0_S3cReT_ON_EThEreuM}'
```

```
> web3.eth.getStorageAt(contractAddress, 2).then(console.log);
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 1409,
  [Symbol(trigger_async_id_symbol)]: 1406,
  [Symbol(destroyed)]: { destroyed: false }
}
> 0x15eea4b2551f0c96d02a5d62f84cac8112690d68c47b16814e221b8a37d6c4d3
```

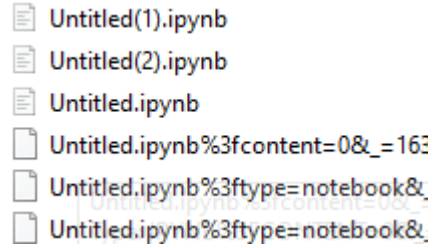
翻车的谜语人

- ~~谜语人翻没翻车我不知道，但我大概在这题翻车了~~
- 这题给了一个pcap包，传统艺能打开wireshark就搜flag1
- 然后发现有什么flag1.txt，一堆HTTP流量，于是先把这些东西都dump出来





翻车的谜语人



- Dump出来的文件中有一堆ipynb，诶好东西那我们打开看一看
- 没法直接显示格式需要稍微调整一下
- 大概是一通操作后把两个随机串存进了flag1.txt和flag2.txt（另一个ipynb中可以看见），于是我们要把他们找出来

```
[26]
import zwsp_steg
from Crypto.Random import get_random_bytes

[27]
import binascii

[28]
def genflag():
    return 'flag{%s}'%binascii.hexlify(get_random_bytes(16)).decode()

[29]
flag1 = genflag()
flag2 = genflag()

key = get_random_bytes(len(flag1))

key

... b'\x1e\xe0[u\xf2\xf2\x81\x01U_\x9d!yc\x8e\xce[X\r\x04\x94\xbc9\x1d\xd7\xf8\xde\xdcd\xb2Q\xa3\x8a?\x16\xe5\x8a9'
```

```
def xor_each(k, b):
    assert len(k)==len(b)
    out = []
    for i in range(len(b)):
        out.append(b[i]^k[i])
    return bytes(out)

encoded_flag1 = xor_each(key, flag1.encode())
encoded_flag2 = xor_each(key, flag2.encode())

with open('flag1.txt', 'wb') as f:
    f.write(binascii.hexlify(encoded_flag1))
```

翻车的谜语人

- Dump出来的文件里直接就有flag1.txt
- 于是稍微算一算拿到第一个flag
- 但并没有flag2.txt只有flag2.7z，看起来还经过了一通操作
- 看一看wireshark的流量统计，发现HTTP已经用了
- 里面剩下可以利用的只有WebSocket了
- 筛出来发现有三条TCP流，7，11，17


```
{
    "name": "flag1.txt",
    "path": "flag1.txt",
    "last_modified": "2021-11-06T07:43:20.952991Z",
    "created": "2021-11-06T07:43:20.952991Z",
    "content": "788c3a1289cbe5383466f9184b07edac6a6b3b37f78e0f7ce79bece502d63091ef5b7087bc44",
    "format": "text",
    "mimetype": "text/plain",
    "size": 76,
    "writable": true,
    "type": "file"
}
```

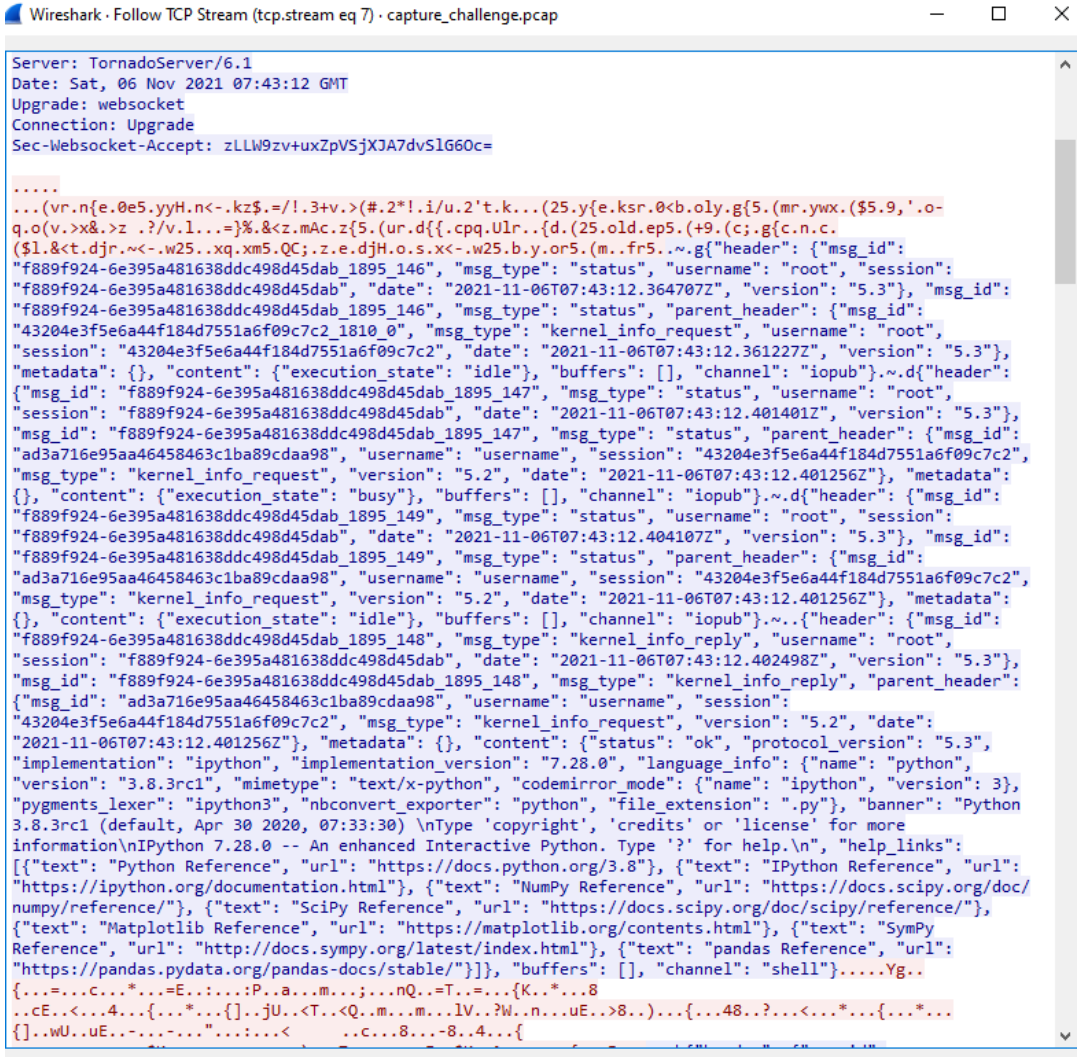

websocket

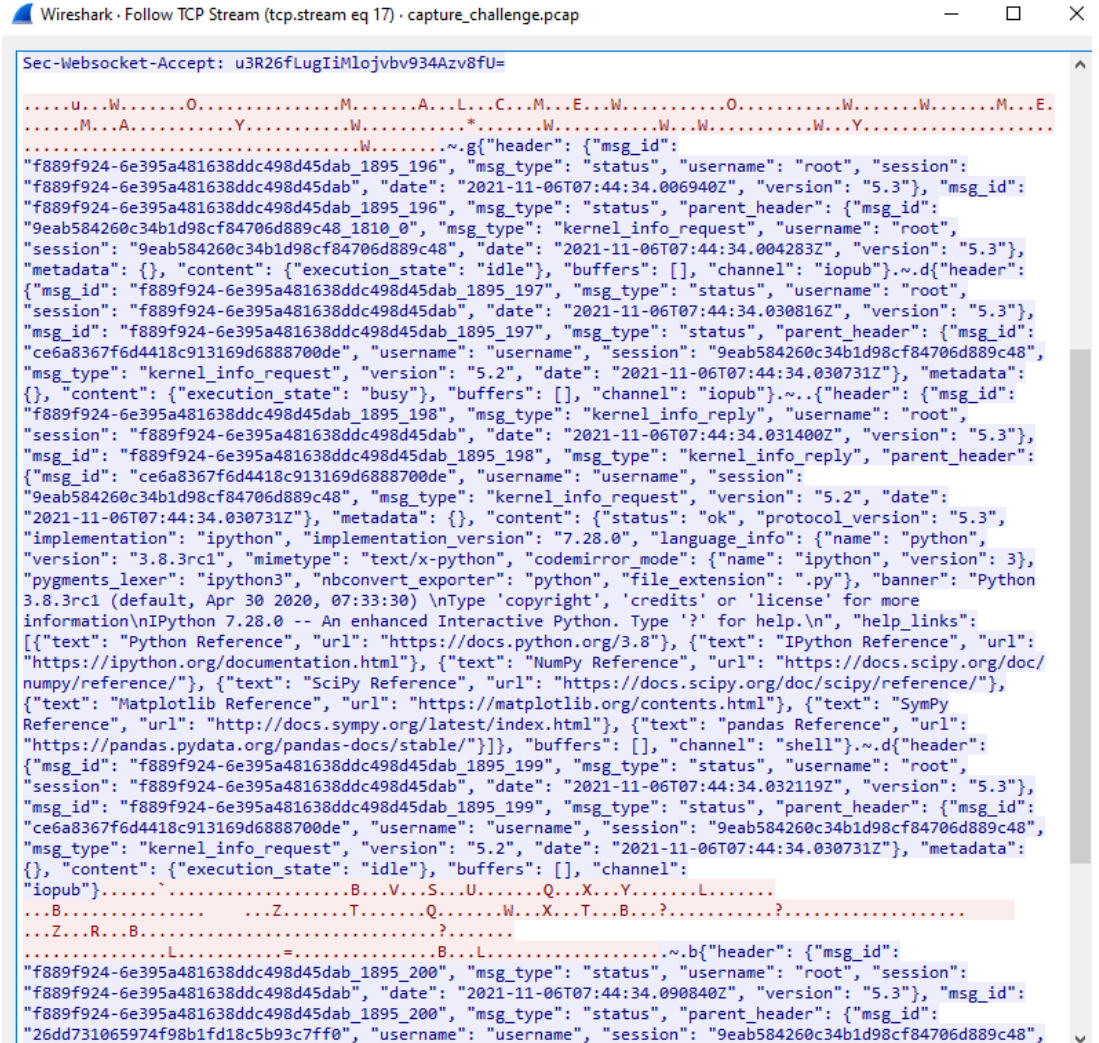
No.	Time	Source	Destination	Protocol	Length	Stream index	Info
5870	8.044766	192.168.17.1	192.168.17.128	WebSoc…	494	7	WebSocket Text [FIN] [MASKED]
5881	8.049702	192.168.17.128	192.168.17.1	WebSoc…	666	7	WebSocket Text [FIN]
5884	8.053850	192.168.17.128	192.168.17.1	WebSoc…	1514	7	WebSocket Text [FIN]
5886	8.054923	192.168.17.128	192.168.17.1	WebSoc…	729	7	WebSocket Text [FIN] WebSocket Text [FI
8450	12.870528	192.168.17.1	192.168.17.128	WebSoc…	494	7	WebSocket Text [FIN] [MASKED]
8451	12.873852	192.168.17.128	192.168.17.1	WebSoc…	666	7	WebSocket Text [FIN]
8452	12.878966	192.168.17.128	192.168.17.1	WebSoc…	1514	7	WebSocket Text [FIN]
8455	12.879209	192.168.17.128	192.168.17.1	WebSoc…	729	7	WebSocket Text [FIN]
9084	14.951518	192.168.17.1	192.168.17.128	WebSoc…	62	7	WebSocket Connection Close [FIN] [MASKE
9086	14.951862	192.168.17.128	192.168.17.1	WebSoc…	58	7	WebSocket Connection Close [FIN]
14340	21.864020	192.168.17.128	192.168.17.1	WebSoc…	167	11	WebSocket Text [FIN]
14360	22.019385	192.168.17.1	192.168.17.128	WebSoc…	88	11	WebSocket Text [FIN] [MASKED]
14362	22.027544	192.168.17.128	192.168.17.1	WebSoc…	162	11	WebSocket Text [FIN]
16485	25.035139	192.168.17.1	192.168.17.128	WebSoc…	73	11	WebSocket Text [FIN] [MASKED]
16487	25.035682	192.168.17.128	192.168.17.1	WebSoc…	71	11	WebSocket Text [FIN]
16508	25.127779	192.168.17.1	192.168.17.128	WebSoc…	73	11	WebSocket Text [FIN] [MASKED]
16510	25.128416	192.168.17.128	192.168.17.1	WebSoc…	71	11	WebSocket Text [FIN]
16558	25.240485	192.168.17.1	192.168.17.128	WebSoc…	73	11	WebSocket Text [FIN] [MASKED]
16559	25.241404	192.168.17.128	192.168.17.1	WebSoc…	71	11	WebSocket Text [FIN]
16692	25.395919	192.168.17.1	192.168.17.128	WebSoc…	73	11	WebSocket Text [FIN] [MASKED]
16693	25.396447	192.168.17.128	192.168.17.1	WebSoc…	71	11	WebSocket Text [FIN]

Protocol	Percent Packets	Packets	Percent Bytes
Frame	100.0	56415	100.0
Ethernet	100.0	56415	1.6
Internet Protocol Version 4	100.0	56392	2.3
User Datagram Protocol	94.7	53449	0.9
Simple Service Discovery Protocol	0.0	4	0.0
Session Traversal Utilities for NAT	1.9	1058	0.2
Dynamic Host Configuration Protocol	0.0	2	0.0
Domain Name System	0.0	8	0.0
Datagram Transport Layer Security	92.8	52377	69.1
Transmission Control Protocol	5.2	2943	25.9
WebSocket	0.8	439	0.0
Line-based text data	0.8	431	0.1
Transport Layer Security	0.3	185	0.3
Hypertext Transfer Protocol	0.4	214	25.4
Portable Network Graphics	0.0	2	0.0
Media Type	0.0	28	23.8
Line-based text data	0.0	15	1.3
JavaScript Object Notation	0.1	38	0.0
Data	0.0	6	0.0
Address Resolution Protocol	0.0	23	0.0

翻车的谜语人

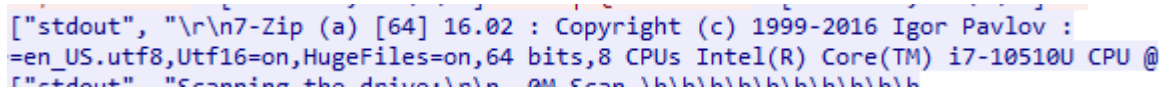
- 分别看一看，stream index为7和17的里面是一同jupyter notebook计算通信
- 看起来没什么用，毕竟结果都在ipynb里面写着呢



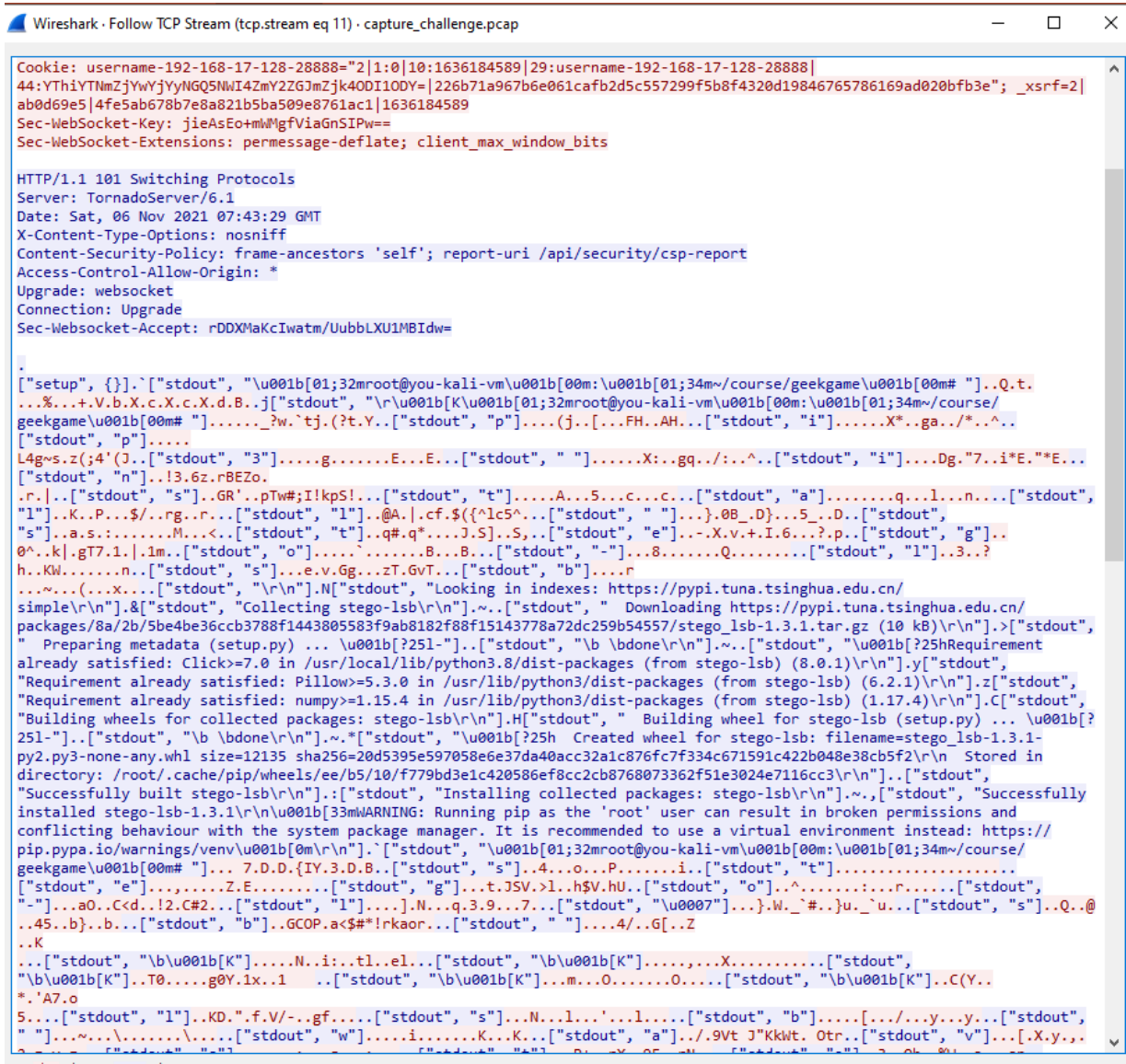


翻车的谜语人

- stream index为11的里面是……诶怎么像是一堆指令p……i……p……3……
- 整理一下，大概完成了一下几条指令：
- pip3 install stego-lsb
- stego-lsb wavsteg -h -i ki-ringtrain.wav -s flag2.txt -o flag2.wav -n 1
- 7za a flag2.7z flag2.wav –p“Wakarimasu! `date` `uname –nom` `nproc`”
- 于是我们只需要逆回去就好了
- nproc可以在最后的通信中看见



- 以及node名称 

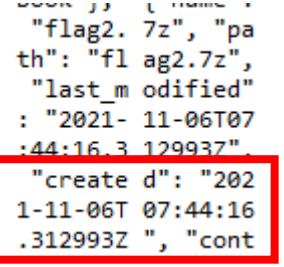


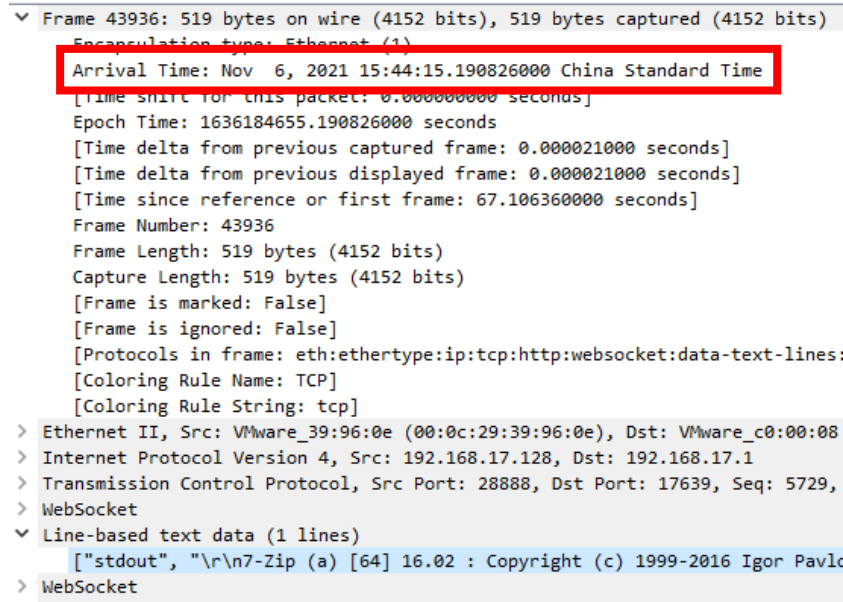
翻车的谜语人

- 剩下的时间`date`和`uname -om`就有点麻烦了，但猜测显示方法和kali一样
- 去找会发现`uname -om`应该是x86_64 GNU/Linux
- `date`是这样的，不知道我的系统为啥不是CST

```
┌──(kali㉿kali)-[~/Desktop]
└─$ date
Wed 17 Nov 2021 11:18:05 PM EST
```

- 然后就得开始猜对面机器的时区了，至于具体的时间可以在log中找到
- 由于要精确到秒，log里有两个时间：
- 一个是**按下回车收到stdout中7za第一条返回消息的时间**
- 另一个是发现**通信中有个目录列表的结果**
- 刚好差了1s，一开始一直选第二个时间
- 导致我以为是别的推错了，后来给了提示
- 确认格式没错后才意识到应该是开始执行指令的
- 时间，所以选第一个，逆向解密后过程同flag1一样





Wakarimasu! Sat 06 Nov 2021 03:44:15 PM CST you-kali-vm x86_64 GNU/Linux 8

叶子的新歌

- **吹爆MaxXing！** 虽然比赛的时候由于分值低和做得人少没先去看，只做了一个flag，但是一个很好玩的题，于是赛后这一天补完了。
- 拿到mp3首先看有没有残留/提示信息

Name	#	Title	Contributing artists	Album
LeafsNewSong.mp3		叶子的新歌	叶子	Secret in Album Cov...

- 等下好家伙……直接显示出来了，那就先按这个做吧

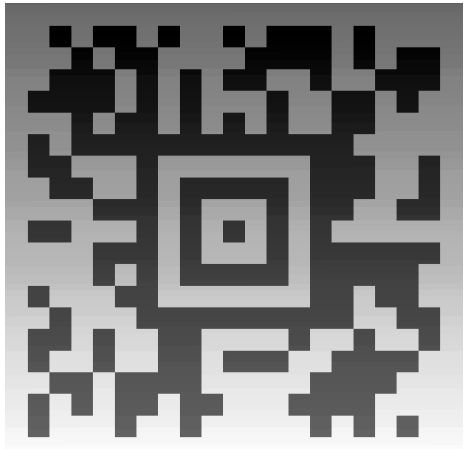
夢は時空を越えて

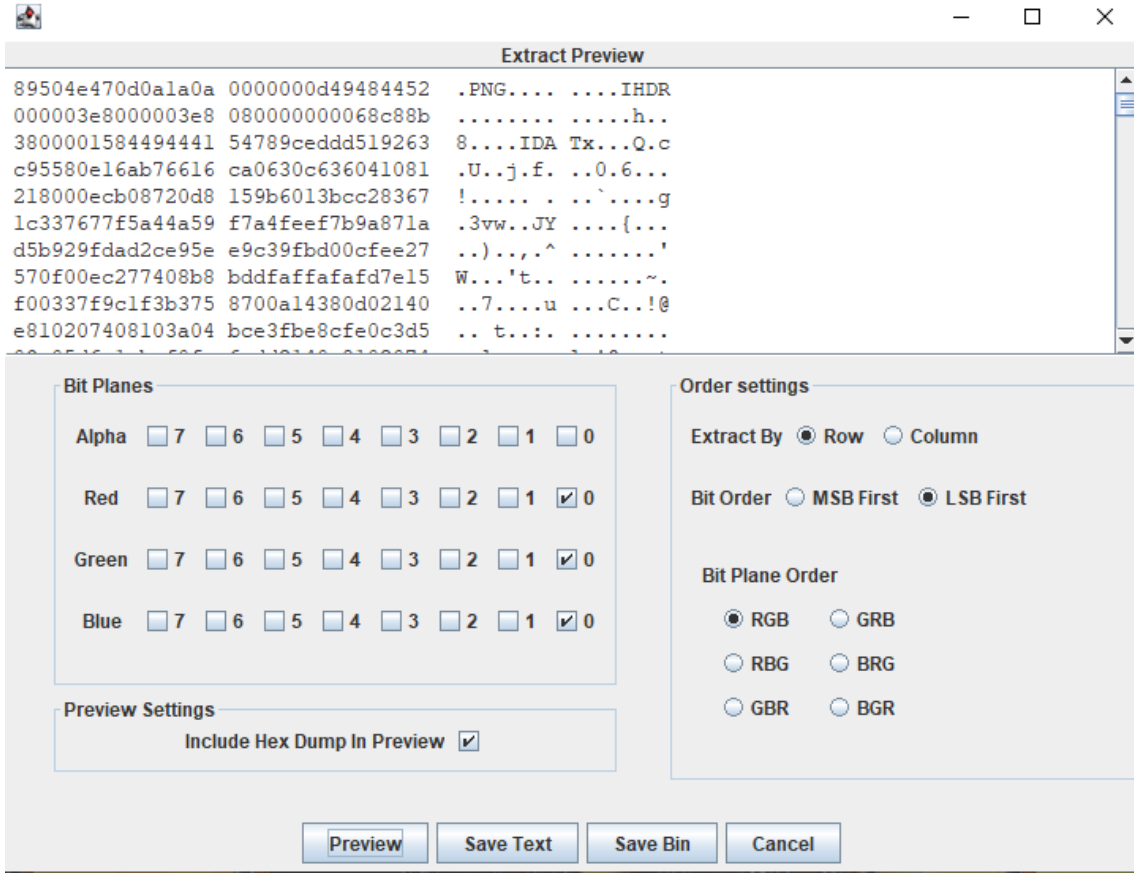
- 由于有**PotPlayer**，直接从里面取出了封面
- 看不出来有什么猫腻，拖到stegsolve里看一眼
- 切换一遍底下的选项都看不出啥
- 于是考虑**lsb隐写**，用data extract功能试一下

叶子的新歌

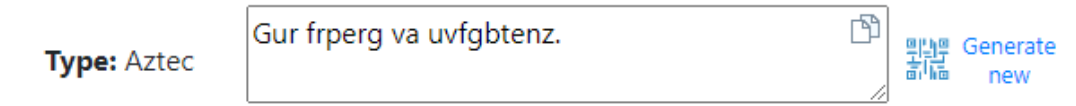
夢は時空を越えて

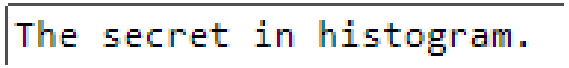
- PNG……快乐了，提取出来看看是什么图片
- 害……原来不是flag
- 于是得到了一张奇怪的
- 像是二维码又不是的图片
- （按照上一届的经历应该
- 识别出来就好了吧）





- Google识图了一下知道这玩意是叫**Aztec Code**找个在线网站识别一下
- 什么玩意……好吧熟练的我知道这是经典
- 凯撒加密了，所以**ROT13**一下得到
- woc……怎么还没结束



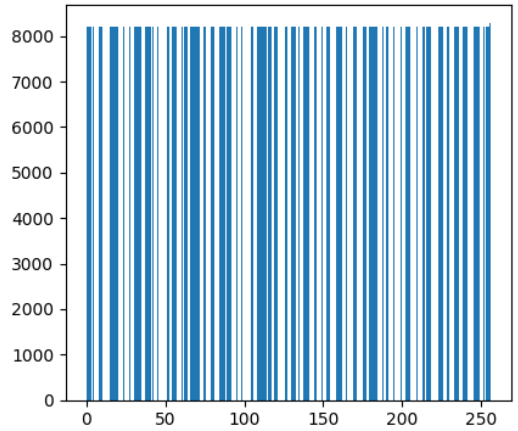


叶子的新歌

夢は時空を越えて

- 这个直方图指谁的呀，在封面上试了一下得到奇怪的东西根本看不懂
- 而且还是彩色的，于是突然想起刚刚的Aztec Code怎么看起来**怪怪的**
- 哦原来就是灰度图而且上面颜色深下面颜色浅呀，拿opencv画个直方图好了


```
[8196, 8196, 8196, 0, 8196, 0, 0, 0, 8196, 8196, 0, 0, 0, 0, 0, 8196, 8196, 8196
, 8196, 8196, 0, 0, 0, 8196, 0, 0, 0, 8196, 0, 0, 8196, 8196, 8196, 8196, 8196,
0, 0, 8196, 8196, 8196, 8196, 0, 8196, 0, 0, 8196, 0, 0, 0, 0, 0, 8196, 8196, 0,
 8196, 8196, 8196, 0, 0, 0, 8196, 0, 8196, 8196, 0, 0, 8196, 8196, 8196, 8196, 8
196, 8196, 0, 0, 8196, 8196, 0, 0, 0, 8196, 8196, 0, 0, 0, 8196, 8196, 8196, 819
6, 0, 8196, 8196, 8196, 0, 0, 0, 8196, 0, 0, 8196, 0, 0, 0, 0, 0, 8196, 8196, 0,
 0, 8196, 8196, 8196, 8196, 8196, 8196, 0, 8196, 8196, 0, 0, 8196, 8196, 0, 0, 0
, 0, 0, 8196, 0, 0, 0, 8196, 8196, 8196, 0, 8196, 0, 0, 8196, 8196, 8196, 8196,
0, 0, 0, 8196, 8196, 0, 0, 0, 8196, 0, 0, 8196, 8196, 0, 0, 0, 0, 8196, 8196, 81
96, 8196, 0, 0, 8196, 0, 0, 0, 0, 8196, 8196, 8196, 0, 0, 0, 0, 8196, 8196, 0, 0, 8196
, 8196, 8196, 8196, 8196, 0, 0, 0, 8196, 0, 0, 8196, 0, 0, 0, 8196, 0, 0, 0, 0,
8196, 0, 0, 8196, 8196, 8196, 0, 0, 0, 0, 8196, 0, 0, 0, 8196, 0, 8196, 8196, 81
96, 0, 0, 0, 0, 0, 8196, 8196, 8196, 0, 0, 8196, 8196, 0, 0, 0, 8196, 8196, 8196
, 0, 0, 8196, 8196, 8196, 0, 0, 0, 0, 8196, 8196, 8196, 8196, 0, 0, 8196, 0, 819
6, 8196, 8284]
```

- 得到一堆意义不明的数据和一张图，这是什么呢，难道8196代表1，其他代表0转换一下就可以得到flag？
- 试了试发现不对全是乱码

叶子的新歌

夢は時空を越えて

- 0101……这不是条形码嘛，我在干啥
- 赶紧拿去扫一扫

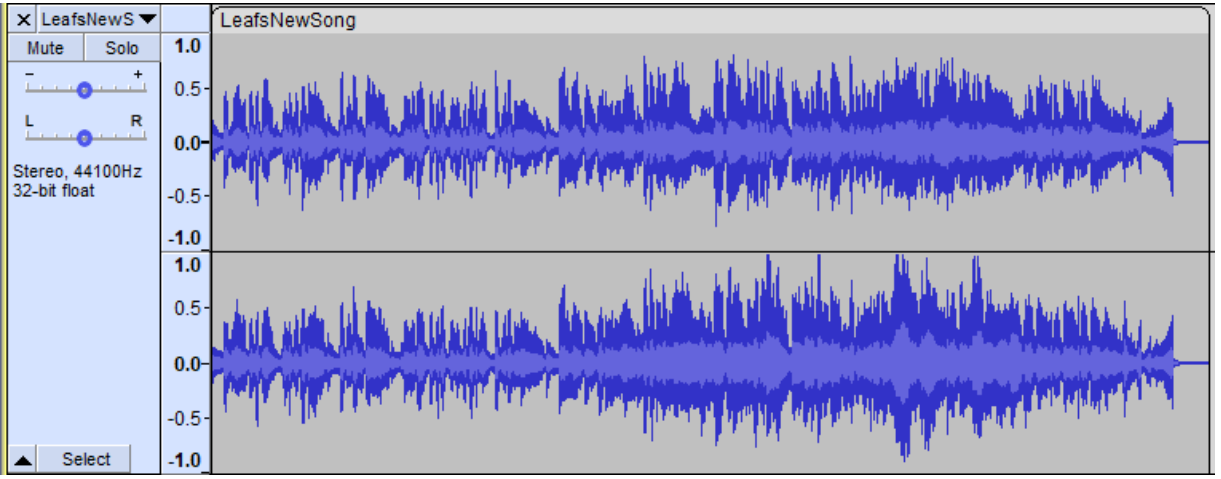
Type: Code128 | xmcp.ltd/KCwBa | Generate new

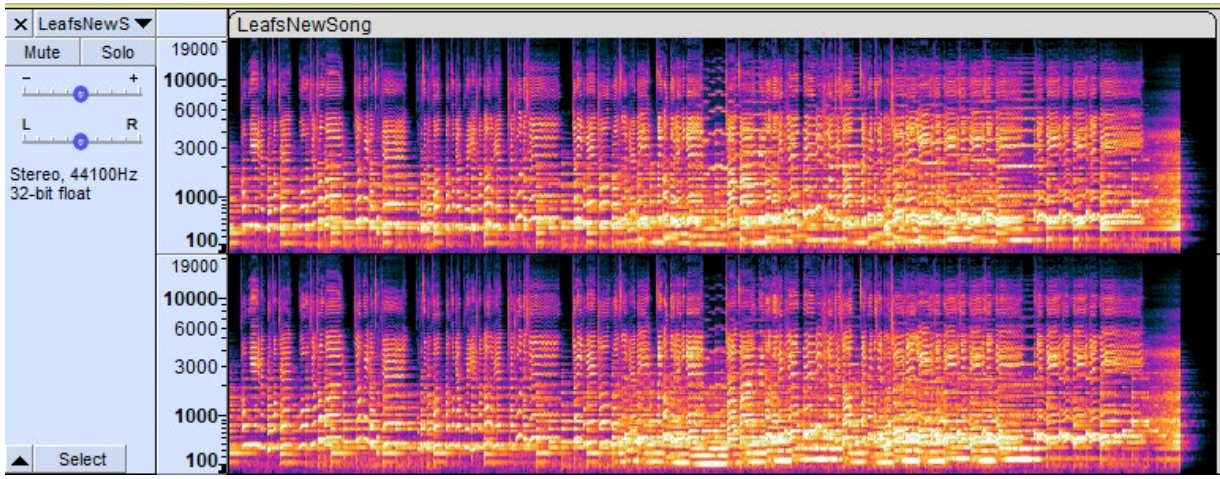
- 好家伙，还没结束
- 看到下面这是一个叫做Ook!的语言（可以说是brainfuck的方言）
- 直接复制运行就好了，结果是
- flag{y0u_h4ve_f0rgott3n_7oo_much}
- ~~（难道这只值200分）~~

叶子的新歌

幻夢界

- 接下来该干什么呢？图片已经用完了，于是我们还剩下音频
- 考虑把音频做一些处理，拖到Audacity里
- 看了看波形、频谱，都没啥异常
- 意料之中，毕竟歌这么好听（雾）





- 于是回去看直接没有认真分析的metadata

叶子的新歌

幻夢界

- 于是回去看直接没有认真分析的metadata，用了ffprobe

```
Metadata:
  TSS             : Logic Pro X 10.7.0
  iTunNORM        :  0000072C 00000736 00003208 00003140 00009E92 0000501A 00006703 00007E86 00007678 00007E1F
  iTunSMPB        :  00000000 00000210 000007A5 00000000002709CB 00000000 002350D1 00000000 00000000 00000000 00000000 00000000 00000000
  title           : 叶子的新歌
  artist          : 叶子
  album           : Secret in Album Cover!!
  TRACKTOTAL      : aHR0cDovL2xhYi5tYXh4c29mdC5uZXQvY3RmL2xlZ2FjeS50Ynoy
  lyrics          : 空无一人的房间
                  : 我望向窗外
                  : 想回到昨天
                  :
                  : 琥珀色的风
                  : 能否将 回忆传到那边
                  : 闪烁的星
                  : 照亮夜空 连成我的思念
                  :
                  : 你 在梦的另一边
                  : 站在 日落的地平线
                  : 背离这世界而去
                  : 想 在回不去的时间里
                  : 遇见你 遇见你 遇见你
                  : 遇见你 遇见你 遇见你
  comment         : 你还记得吗？小时候，我家和你家都在一个大院里。放学以后，我们经常一起在院子里玩。你虽然是个女孩子，但总是能和男孩子们玩到一块去。
                  :
                  : 夏天的时候我们挖蚯蚓、捉蚂蚱；冬天，院子里的大坡上积了一层雪，我们就坐在纸箱子压成的雪橇上，一次次从坡顶滑到坡底。那个时候你还发现，坐在铁簸箕上滑得更快。
                  :
                  : ——当然，那次你也摔得挺惨的。
  encoder         : Lavf58.45.100
```

- 可以看出：苹果电脑上做的、有歌词&文案，album那里用过的提示和……

叶子的新歌

幻夢界

- 这个TRACKTOTAL什么意思呀？

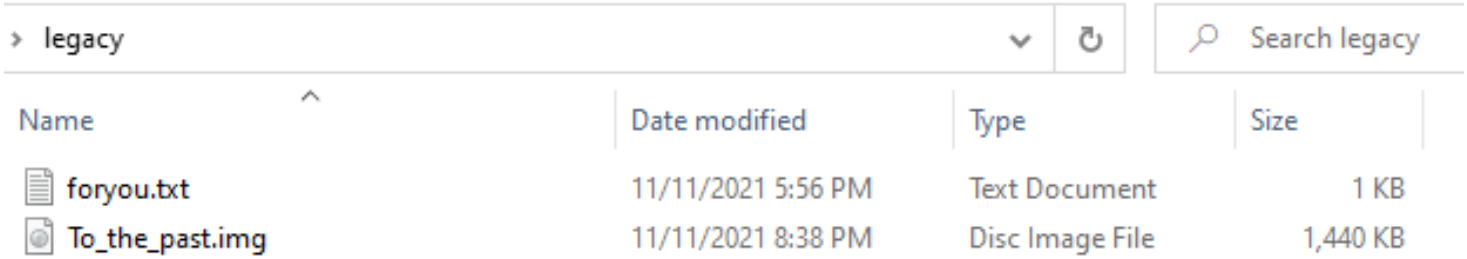
```
TRACKTOTAL      : aHR0cDovL2xhYi5tYXh4c29mdC5uZXQvY3RmL2xlZ2FjeS50Ynoy
```

- 音轨数？怎么会这么多，而且看着像什么加密文本，拿去base64 decode下

Base64 编码或解码的结果：

http://lab.maxxsoft.net/ctf/legacy.tbz2

- 好家伙，又是一个出题人的网站，下载后解压一下得到



- 我们看看里面都是什么

叶子的新歌

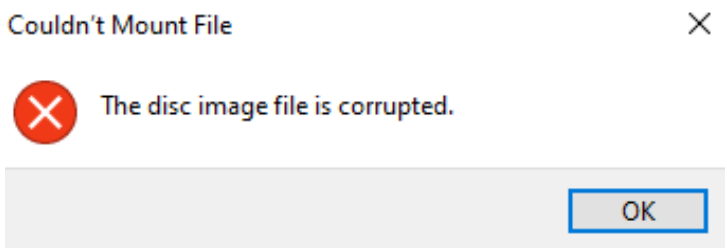
*疑似neta

幻夢界

- foryou.txt里面是

> 我有一张很久很久以前的**软盘**。说起来以前的操作系统还能装在**软盘**里，把**软盘**放进电脑就可以启动，很神奇吧？
> 我给你看过这张软盘，但你总说这是Word保存图标的手办……什么跟什么啦！
>
> 现在已经没有**带软驱的电脑**了，甚至连带光驱的电脑都没有了。以前**软盘**里的那些东西，也许再也启动不了了吧。
>
> 时间过得好快啊，转眼间，就来到了现实。

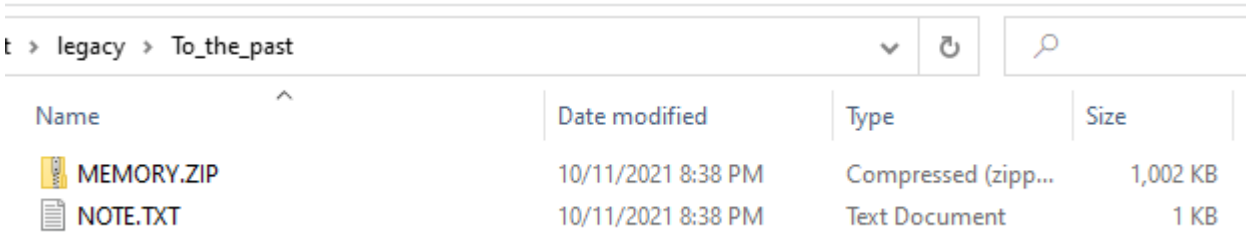
- 另一个是.img文件，懂了这个就是软盘里的东西
- 打开看看是啥，试着直接加载一下



- 好吧，看起来不行，但用7zip解压一下发现成功了

叶子的新歌

幻夢界



- 于是我们得到了
- NOTE.TXT里面是

> 备忘
>
> 密码是：宾驭令诠怀驭榕喆艺艺宾庚艺怀喆晾令喆晾怀

- 另一个是ZIP，懂了直接解压就好了……密码错误！
- 果然这一串看起来很奇怪的密码不行
- 到网上查一下发现这个是第五套人民币钱币箱号和冠号的一个单表加密

艺=1 驭=2 令=3 怀=4 庚=5 诠=6 宾=7 晾=8 喆=9 榕=10也就是0

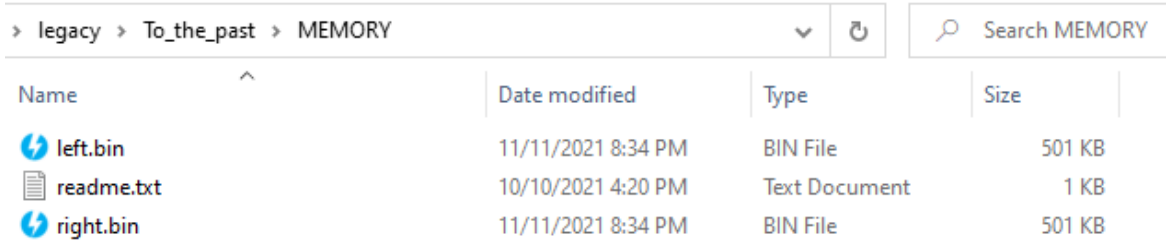
- 翻译得到72364209117514983984

叶子的新歌

*疑似neta

幻夢界

- 我们这都进了几层了……继续解压得到



- readme.txt里面写着

我以前很喜欢玩**红白机**，当然，现在也很喜欢。超级马里奥、魂斗罗、坦克大战、马戏团、冒险岛……

一玩能玩一天。小时候家里有一台**红白机**，也经常叫你一起玩游戏，只不过，我记得你不喜欢这些东西。你最喜欢在4399玩**找不同**，而且你还玩的特别棒，简直就是**找不同**滴神。

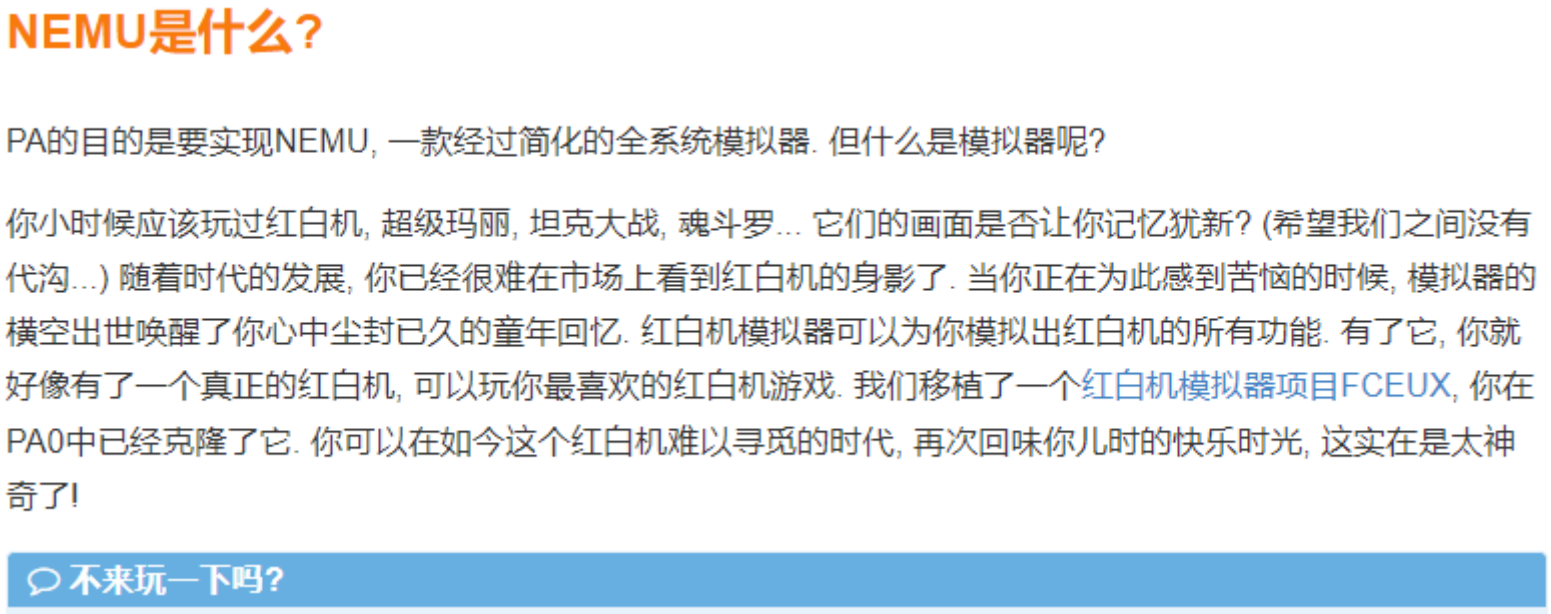
呜呜，**红白机**已经属于时代的眼泪了。

- 所以.bin文件是游戏文件？……而且这时候熟悉第一句话的同学可能会发现

叶子的新歌

幻夢界

- https://nju-projectn.github.io/ics-pa-gitbook/ics2021/1.1.html



NEMU是什么?

PA的目的是要实现NEMU, 一款经过简化的全系统模拟器. 但什么是模拟器呢?

你小时候应该玩过红白机, 超级玛丽, 坦克大战, 魂斗罗... 它们的画面是否让你记忆犹新? (希望我们之间没有代沟...) 随着时代的发展, 你已经很难在市场上看到红白机的身影了. 当你正在为此感到苦恼的时候, 模拟器的横空出世唤醒了你心中尘封已久的童年回忆. 红白机模拟器可以为你模拟出红白机的所有功能. 有了它, 你就好像有了一个真正的红白机, 可以玩你最喜欢的红白机游戏. 我们移植了一个红白机模拟器项目FCEUX, 你在PA0中已经克隆了它. 你可以在如今这个红白机难以寻觅的时代, 再次回味你儿时的快乐时光, 这实在是太神奇了!

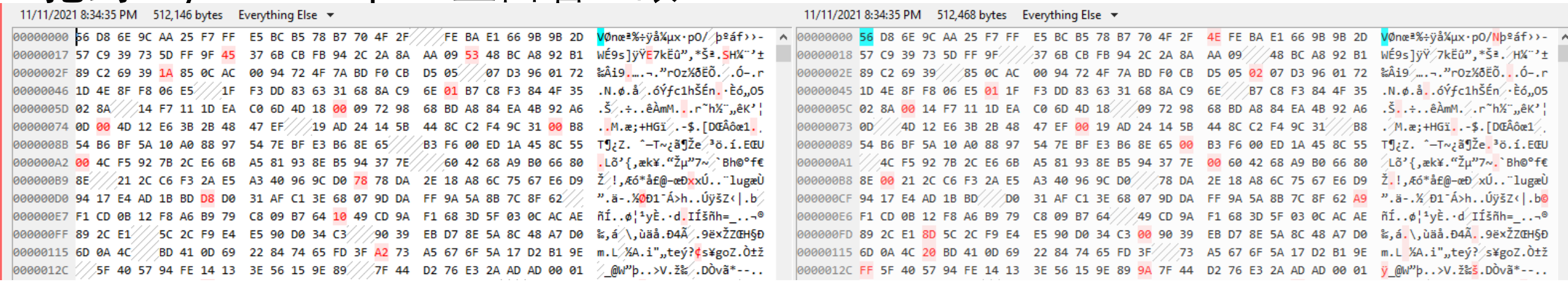
不来玩一下吗?

- 可以更加确信我们的想法

叶子的新歌

幻夢界

- 那么是哪种游戏呢，上网查一下发现.bin文件很可能是世嘉的游戏
- 下载Kega Fusion发现不管是left.bin还是right.bin好像都跑不动
- 说起来总感觉文本里有什么词没用上——**“找不同”**
- 再联系这个文件名left/right这是指左右两张图吗？
- 拖到Beyond Compare里面看一眼

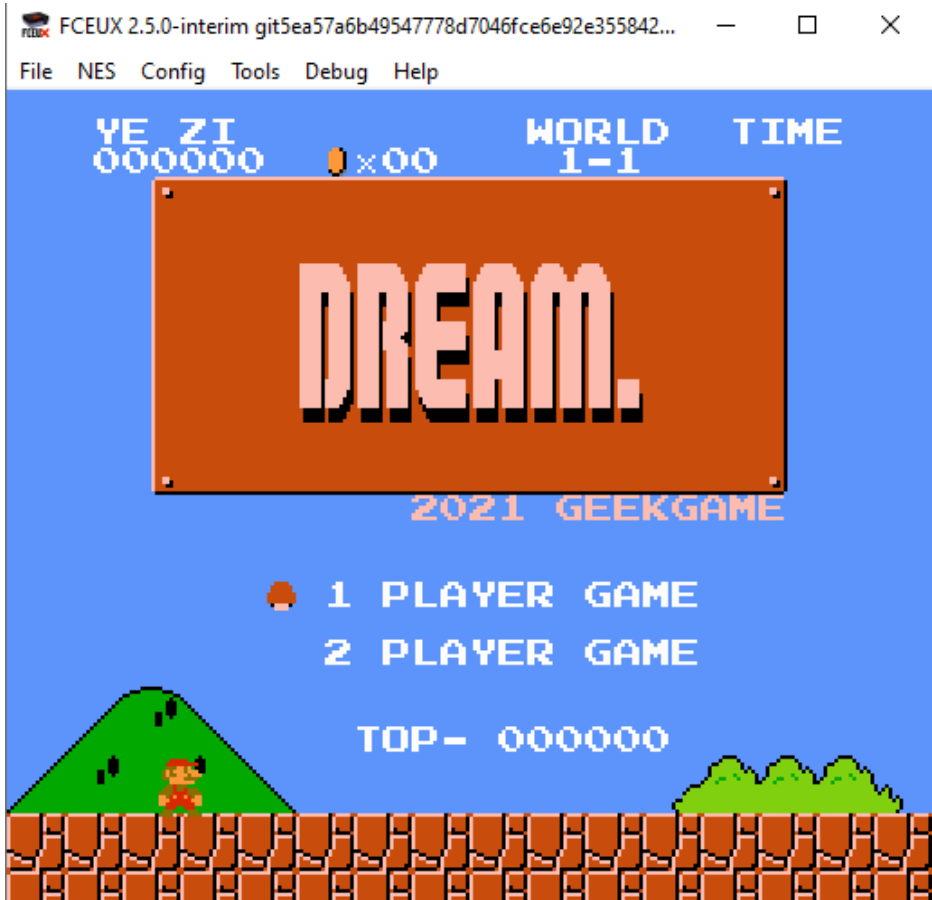


- 好乱啊，这是什么，等下……NES……这不就是指红白机嘛

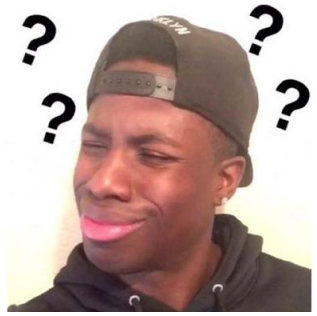
叶子的新歌

幻夢界

- 于是通过一通命令操作把差异部分取出来得到游戏game.nes
- 下载了一个叫做**fceux**的模拟器试着玩一玩，里面还有Debugger
- 名字变成了叶子(YE ZI)，标题是(DREAM.)
- 嗯符合剧情，那游戏里有啥猫腻呢？
- 手打了几关，发现……这和原版一模一样
- 于是猜测可能通关才能得到flag
- 看看怎么魔改这个文件来作弊

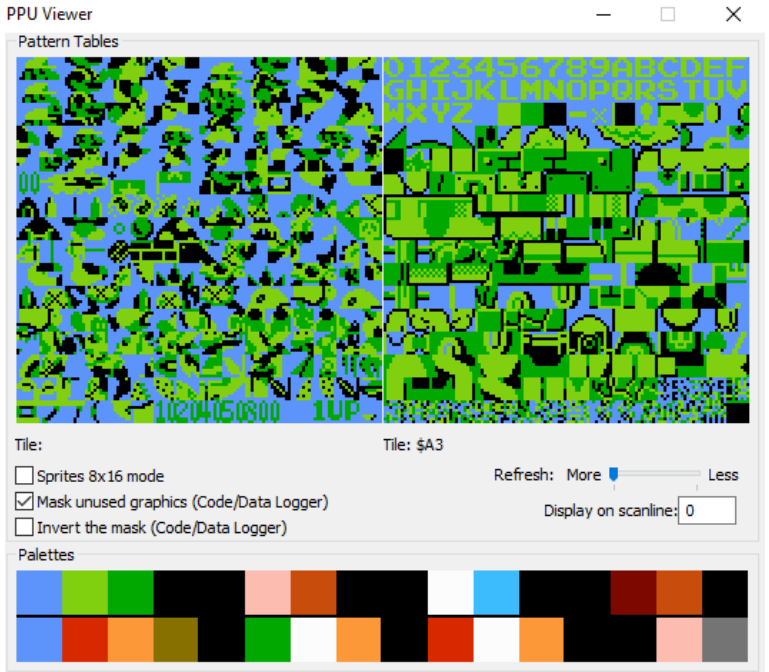


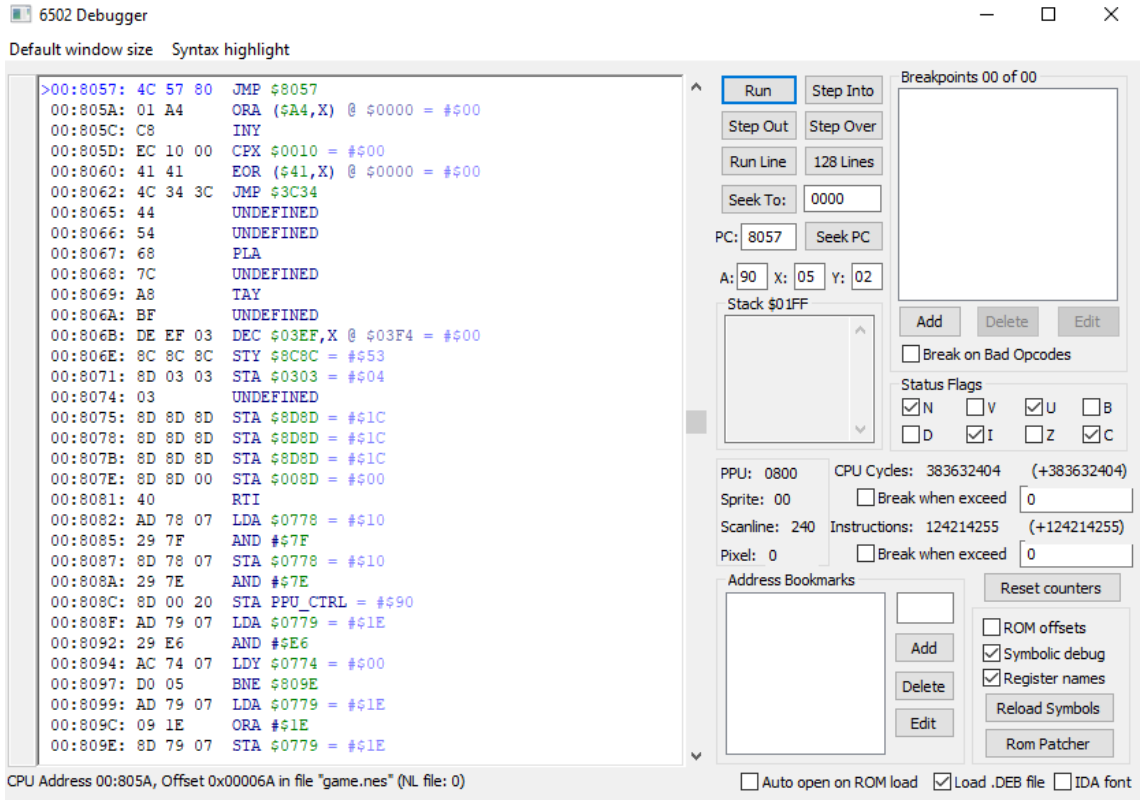
叶子的新歌



幻夢界

- 打开Debugger，发现PC直接死循环了
- 如果有人能告诉我发生了什么就好，但总之放弃了这个想法
- 接着发现有个图库，根据这里的二进制找到了标题页的YEZI在内存的位置
- 那么我们试着找下flag
- 好吧，果然没有
- ……

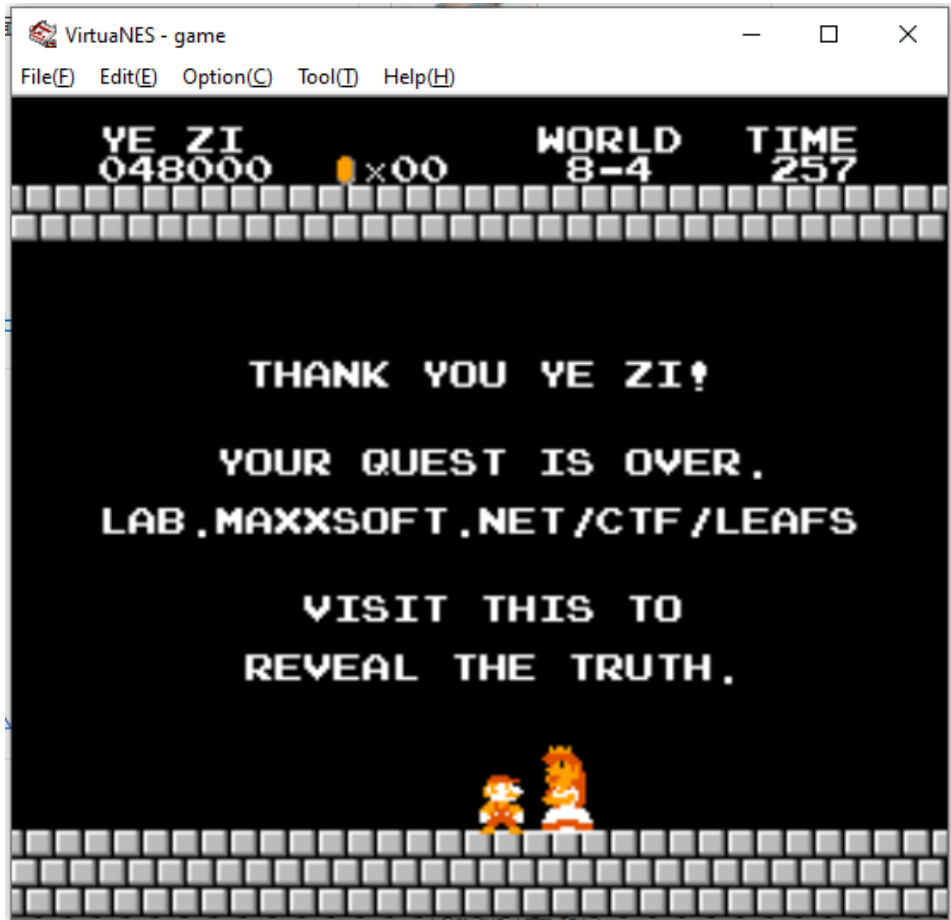




叶子的新歌

幻夢界

- 那我们直接打游戏吧，当然打是不可能自己打的（老手残了）
- 于是又下载了**VirtualNES**，同时在网上找到了金手指：
- 079F-01-09 金身无敌
- 0704-01-01 飘浮（如在水中一般）
- 075F-01-07 选大关
- 0760-01-03 选小关
- 直接跳到8-4，开着无敌打游戏，好耶
- 通关得到了一个网址……怪不得搜flag搜不到



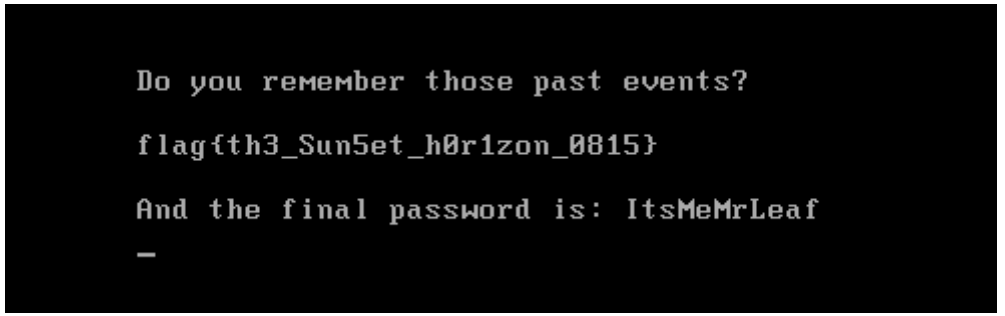
叶子的新歌

幻夢界

- 打开是一个孤零零的页面
- 什么软盘？什么密码？
- 哦刚刚第一步压缩包就提到了软盘，难道有什么东西漏掉了？
- 回去看下那个刚刚加载失败的img文件

```
$ file To_the_past.img
To_the_past.img: DOS/MBR boot sector, code offset 0x3c+2, OEM-ID "GeekGame", root entries 224, sectors 2880 (volumes <=32 MB), sectors/FAT 9, sectors/track 18, serial number 0x0, label: "TO_THE_PAST", FAT (12 bit), followed by FAT
```

- DOS/MBR boot？难道是个启动盘
- 于是丢到VMware里试一下



- 好家伙，原来img里就有一个flag{th3_Sun5et_h0r1zon_0815}

叶子的新歌

幻夢界 & 夢と現の境界

- 密码ItsMeMrLeaf大概就是网页需要的吧
- 输进去得到最后一个flag
- flag{W4ke_up_fr0m_7h3_L0NG_dre@m}

- 原来是之前没注意到软盘
- 这就是为什么做出来的人都是
- 几乎后两个同时交的原因吧

说了那么多的回忆，不如，再来聊聊现在吧。

那天和你表白之后，我们正式在一起了。回想起那段日子，整个世界都是彩色的，在校园里的每一天，我和你都很开心。

但，如果你没有在8月15号出去练琴，如果你没有经过那个十字路口，如果那辆卡车没有超速。

如果。

如果这个世界上真的有如果。

从那之后到现在，你已经昏迷整整三年了。

大夫说，你失去了所有的记忆，再也想不起我们的从前，变成了一个无意识的植物人。

但我依然相信，只要我每天都在床边陪着你，和你讲以前的故事，奇迹就一定会发生。

如果你在那个悠长的梦里，看到了这条消息，请一定记得：

从你的世界里醒过来，然后——

回家吧。

flag{W4ke_up_fr0m_7h3_L0NG_dre@m}

叶子的新歌

歌词

空无一人的房间

我望向窗外

想回到昨天

琥珀色的风

能否将 回忆传到那边

闪烁的星

照亮夜空 连成我的思念

你 在梦的另一边

站在 日落的地平线

背离这世界而去

想 在回不去的时间里

遇见你 遇见你 遇见你

遇见你 遇见你 遇见你

剧情汇总

你还记得吗？小时候，我家和你家都在一个大院里。放学以后，我们经常一起在院子里玩。你虽然是个女孩子，但总是能和男孩子们玩到一块去。

夏天的时候我们挖蚯蚓、捉蚂蚱；冬天，院子里的大坡上积了一层雪，我们就坐在纸箱子压成的雪橇上，一次次从坡顶滑到坡底。那个时候你还发现，坐在铁簸箕上滑得更快。

——当然，那次你也摔得挺惨的。

你还记得高中的时候吗？那时在市里的重点中学，我们是同桌。我以前还怪讨人嫌的，老是惹你生气，然后你就不和我说话，我就死乞白赖地求你，或者讲笑话逗你。 不过，你笑起来好可爱，从小就好可爱。此后的一切，也都是从那个笑容开始的吧。 真的，好想回到那个时候啊。

说了那么多的回忆，不如，再来聊聊现在吧。

我有一张很久很久以前的软盘。说起来以前的操作系统还能装在软盘里，把软盘放进电脑就可以启动，很神奇吧？我给你看过这张软盘，但你总说这是Word保存图标的手办……什么跟什么啦！现在已经没有带软驱的电脑了，甚至连带光驱的电脑都没有了。以前软盘里的那些东西，也许再也启动不了了吧。时间过得好快啊，转眼间，就来到了现实。

我以前很喜欢玩红白机，当然，现在也很喜欢。超级马里奥、魂斗罗、坦克大战、马戏团、冒险岛……

一玩能玩一天。小时候家里有一台红白机，也经常叫你一起玩游戏，只不过，我记得你不喜欢这些东西。你最喜欢在4399玩找不同，而且你还玩的特别棒，简直就是找不同滴神。

呜呜，红白机已经属于时代的眼泪了。

那天和你表白之后，我们正式在一起了。回想起那段日子，整个世界都是彩色的，在校园里的每一天，我和你都很开心。

但，如果你没有在8月15号出去练琴，如果你没有经过那个十字路口，如果那辆卡车没有超速。

如果。

如果这个世界上真的有如果。

从那之后到现在，你已经昏迷整整三年了。

大夫说，你失去了所有的记忆，再也想不起我们的从前，变成了一个无意识的植物人。

但我依然相信，只要我每天都在床边陪着你，和你讲以前的故事，奇迹就一定会发生。

如果你在那个悠长的梦里，看到了这条消息，请一定记得：

从你的世界里醒过来，然后——

回家吧。

在线解压网站

- 给了源码，读一下发现
- 它还真就直接把上传
- 的压缩包原样解压访问
- 那么问题就简单了：
- 构造一个软链接指向/flag
- 打包传上去就好了
- 得到flag：
- flag{NeV3r_trUST_Any_C0
- mpresSEd_File}

```python
@app.route('/',methods=['POST', 'GET'])
def index():
    if request.method == 'POST':
        f=request.files['file']
        os.system("rm -rf /dev/shm/zip/media/*")
        path=os.path.join("/dev/shm/zip/media",'tmp.zip')
        f.save(path)
        os.system('timeout -k 1 3 unzip /dev/shm/zip/media/tmp.zip -d /dev/shm/
        os.system('rm /dev/shm/zip/media/tmp.zip')
        return redirect('/media/')
    response = render_template('index.html')
    return response

@app.route('/media/',methods=['GET'])
@app.route('/media',methods=['GET'])
@app.route('/media/<path>',methods=['GET'])
def media(path=""):
    npath=os.path.join("/dev/shm/zip/media",path)
    if not os.path.exists(npath):
        return make_response("404",404)
    if not os.path.isdir(npath):
        f=open(npath,'rb')
        response = make_response(f.read())
        response.headers['Content-Type'] = 'application/octet-stream'
        return response
    else:
        fn=os.listdir(npath)
        fn=[".."]+fn
        f=open("templates/template.html")
        x=f.read()
        f.close()
        ret="<h1>文件列表:</h1><br><hr>"
        for i in fn:
            tpath=os.path.join('/media/',path,i)
            ret+="<a href='"+tpath+"'>"+i+"</a><br>"
        x=x.replace("HTMLTEXT",ret)
        return x
```

Flag即服务

- 一开始没做出来，后来根据提示补了flag1
- 首先没给源码，开始乱尝试，发现路径为/api/的时候会报错

```
Error: EISDIR: illegal operation on a directory, read
    at Object.readSync (fs.js:617:3)
    at tryReadSync (fs.js:382:20)
    at Object.readFileSync (fs.js:419:19)
    at /usr/src/app/node_modules/jsonaas-backend/index.js:56:19
    at Layer.handle [as handle_request] (/usr/src/app/node_modules/express/lib/router/layer.js:95:5)
    at next (/usr/src/app/node_modules/express/lib/router/route.js:137:13)
    at Route.dispatch (/usr/src/app/node_modules/express/lib/router/route.js:112:3)
    at Layer.handle [as handle_request] (/usr/src/app/node_modules/express/lib/router/layer.js:95:5)
    at /usr/src/app/node_modules/express/lib/router/index.js:281:22
    at param (/usr/src/app/node_modules/express/lib/router/index.js:354:14)
```

- 注意到这个jsonaas-backend就是题目的名字，所以这是源码的位置
- 至于怎么取出来，用和上一题类似的方法+提示中的package.json

```
curl --path-as-is https://prob11-ce663zj8.geekgame.pku.edu.cn/api/../node_modules/jsonaas-backend/package.json
{"_from":"https://geekgame.pku.edu.cn/static/super-secret-jsonaas-backend-1.0.1.tgz","_id":"jsonaas-backend@1.0.1","_inBundle":false,"_integrity":"sha512-1QXyB4EMI5AyDxZKBKd67uKv6ih4WmLElayHqh/PVo/L1JzSN1zWdPJkzch92OGAE2uNd8udoHLYXNrIXKIf9A==","_location":"/jsonaas-backend","_phantomChildren":{},"_requested":{"type":"remote","raw":"jsonaas-backend@https://geekgame.pku.edu.cn/static/super-secret-jsonaas-backend-1.0.1.tgz","name":"jsonaas-backend","escapedName":"jsonaas-backend","rawSpec":"https://geekgame.pku.edu.cn/static/super-secret-jsonaas-backend-1.0.1.tgz","saveSpec":"https://geekgame.pku.edu.cn/static/super-secret-jsonaas-backend-1.0.1.tgz","fetchSpec":"https://geekgame.pku.edu.cn/static/super-secret-jsonaas-backend-1.0.1.tgz"},"_requiredBy":["/"],"_resolved":"https://geekgame.pku.edu.cn/static/super-secret-jsonaas-backend-1.0.1.tgz","_shasum":"9af4bb380e83a63b7e04e0edf8482bb00b2f9f35","_spec":"jsonaas-backend@https://geekgame.pku.edu.cn/static/super-secret-jsonaas-backend-1.0.1.tgz","_where":"/usr/src/app","author":{"name":"You"},"bundleDependencies":false,"dependencies":{"express":"^4.17.1","express-session":"^1.17.2","helmet":"^4.6.0"},"deprecated":false,"description":"","license":"WTFPL","main":"index.js","name":"jsonaas-backend","scripts":{},"version":"1.0.1"}
```

- 下载源码之后，发现第一个flag是

```
> `flag{${0.1+0.2}}`
< 'flag{0.30000000000000004}'
```

诡异的网关

- 打开发现这里好像记录了密码，根据名字这就应该是flag
- 试着删除它发现外面的config文件变小了
- 逻辑大概是从里面读保存的用户名密码，但打开看一眼发现应该是加过密的
- 但程序运行的时候读进去应该需要解密
- 于是使用Ollydbg在运行的时候查内存发现有一块是刚好就是flag{……}
- 这题忘了截图了……

最强大脑

- 读IDA反编译的代码发现逻辑是
- 程序先把flag1拷到brainfuck需要的数据数组的末端（刚开始大小4096）
- 然后模拟brainfuck程序
- 唯一的问题在于程序长度限制4096
- 而要移动数据指针4096次
- 策略是:
- 考虑到数组数组会清零，于是向右一直找到第一个f（brainfuck的循环是检测是否为0停止的，于是对每个位置-102即可），接着把后面的打印十几二十位出来即可
- 程序是：.----------[>.----------]>.
- 这样就可以得到第一个flag：flag{N0_traIning_@_@LI}

密码学实践

```
for it in range(0,len(mess),32):
    pmess=mess[it:it+32]
    a = bytes_to_long(pmess[0:8])
    b = bytes_to_long(pmess[8:16])
    c = bytes_to_long(pmess[16:24])
    d = bytes_to_long(pmess[24:32])
    for key in keys:
        a, b, c, d = b, c, d, a ^ c ^ key
    a=long_to_bytes(a,8)
    b=long_to_bytes(b,8)
    c=long_to_bytes(c,8)
    d=long_to_bytes(d,8)
    cip+=a+b+c+d
return cip
```

flag1

- 读程序发现问题在于MESenc这个函数有问题：
- 其中这部分加密，a,b,c,d在经过运算后的结果
- 一定都是a,b,c,d和所有key的异或组合中的一种
- 并且无论a,b,c,d取何值，key的部分的异或组合是不变的
- 这个是可以通过一组加密前后的数据计算出来的
- 即若从a,b,c,d得到x,y,z,w，而把^key的部分去掉再算一次得到x’,y’,z’,w’
- 把原始加密函数记作A，去掉^key的记作B
- 那么x^x’, y^y’, z^z’, w^w’就是为了算出A四部分分别要在B结果上异或的值
- 那么通过Richard的第二条明文密文信息先算出这个，再作用到第一条密文上即可得到第一条明文原始消息了

密码学实践

flag2

- 读程序发现问题在于God能告诉你同一组N和d加密多个证书（<=3）的结果
- 那么我们通过模意义下乘法就可以伪造新的数据作为证书
- 读doRichard的代码知道我们只要伪造一个(Alice,0)的证书就可以用flag1的方法解出来了
- 方法是:
- 由于(Alice,0)=b‘Alice\x00\x05\x00\x00’
- 第一次name=空，key=‘416c696365’
- 第二次name=‘00’ ,key=空
- 把两个乘起来就是(Alice,0)的证书了

扫雷

- 读程序发现扫雷就是个壳，本质问题是，
- 给一个随机数生成器，每次生成一组8个32bit数，需要你全部猜对
- 困难模式是直接按顺序连续给出，每次给一组8个32bit数
- 简单模式是中间可能空跳几轮，就是说每次拿到8个32bit数，但你不知道这次和上次直接有没有漏掉几组8个32bit数，每组数都以1/2的概率丢失

困难模式

- 分析Python的梅森旋转算法https://liam.page/2018/01/12/Mersenne-twister/
- 可以知道从最初开始，每624数一组决定了内部的状态，因此只要得到624个开头开始连续的数就可以还原内部状态从而预测之后的值

简单模式

$$\vec{x}_{k+n} \stackrel{\text{def}}{=} \vec{x}_{k+m} \oplus \left(\vec{x}_k^{(u)} \mid \vec{x}_{k+1}^{(l)}\right)\mathbf{A}.$$

- 再仔细看公式会发现可以通过第0,1,397个数推测第624个数，同时由于1/2的概率，直接猜测第25/26组对应397的位置，第40组数对于第624个数开始的8个我们要猜的二进制数。这样尝试成功的概率经测试大概是(1~2)/500，多试几次足够用来解决本题