

第 1 届 GeekGame 参赛 WriteUp

昵称: wwzzj (汪汪吱吱叽)

签到

文件夹: signin

这道题打开之后是一个 pdf 文档, 下面的乱码应该就是 flag 的某种形式了, 直接复制发现复制不全。

于是用 Acrobat 的编辑模式将内容复制了出来。

```
fa{aeAGetTm@ekaev!  
lgHv__ra_ieGeGm_1}
```

发现是竖着排列的 flag, 写了几行 python 调整格式, 得到答案!

小北问答 Remake

文件夹: QA

第一题: 共有几号理科楼, 凭借生活经验回答, 5

第二题, 上届比赛注册人数, 从 linux 俱乐部的公众号上找了, 407

第三题: geekgame 网站上一次 https 证书过期, 这题我找了好久, 不停变换关键词搜索, 希望找到一个有 https 证书记录的网站, 结果最后还是在 google 上通过中文搜到的:

关键词: https 证书 历史

<https://pc6a.com> > ... · [Translate this page](#) ⋮


分享一个SSL证书签发历史查询网站

Mar 8, 2018 – 网址: <https://crt.sh/> 可以查询自己网站什么时候, 通过谁签发了SSL证书。我就查到一个COMODO, 莫名其妙的签发历史。

可能是我的英文搜索技巧不够地道, 或者是外国人没有这种需求, 怎么用英文怎么搜都搜不到。2021-07-11T08:49:53+08:00

第四题: DEFCON 签到题的 flag, 在网站 <https://scoreboard2020.ooverflow.io/#/> 上, 第一道题的 flag 竟然是点击就送。OOO{this_is_the_welcome_flag}



第五题: 有嘉心糖在引流!  , 在 google 上一搜全是 asoul 的内容! 最后没有找到现成的答案。

使用恰好有几对皇后会互相攻击推了一下容斥的式子, 发现计算恰好有两对时化简比较麻烦, 感觉写了个 O(边长)的程序, 结果怎么都不对。然后写了个暴力对拍, 终于调对了。

第二阶段时被提醒可以用 <https://www.wolframalpha.com/> 直接算, 感觉白白浪费了很多时间, 亏豹! 2933523260166137923998409309647057493882806525577536

第六题: 看 github 上去年比赛的源码。Submits

第七题：这题我一直以为是 AS24349 CERNET2 IX at Peking University，而且深信不疑，最后交了好多次才定位原来是这道题错了，重新搜，才发现原来还有 AS59201 Peking University in China。

第八题：信息官网上 2021 年 6 月 3 日有个招生简章，上面列了所有的实验室。找到最长的就行。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

这道题我也花了很多时间，竟然放得这么靠前，做出来的人这么多，看了一晚上连猜带蒙才做出来。

之前没有接触过智能合约完全不知道要做什么，查了发现智能合约上可以放代码，一开始以为要交互，按照说明配置 **Remix** 和 **MetaMask**，但一直跑不起来。

然后去看反编译的代码，**Etherscan** 和 **Ethervm.io** 上的竟然还不一样，不过看起来 Ethervm.io 要专业一些，看来代码大概知道是位运算操作，看了看交易记录发现只有一条 **Create**，还以为管理员没有设置参数，所以两个参数都是默认的 **0**，直接逆着位运算算了一下，结果怎么都不对。

尝试了好久在网上查到一个 Writeup，竟然还能在 Etherscan 上通过 **parity trace** 查看交易的输入和输出，然后才发现还有一条 Internal 交易，感觉输入应该是设置两个参数的，但是位数不对，猜他的编码方式是前面几位是函数名，后面两个 256 位的整数，尝试之后竟然真的解出了 flag。

翻车的谜语人

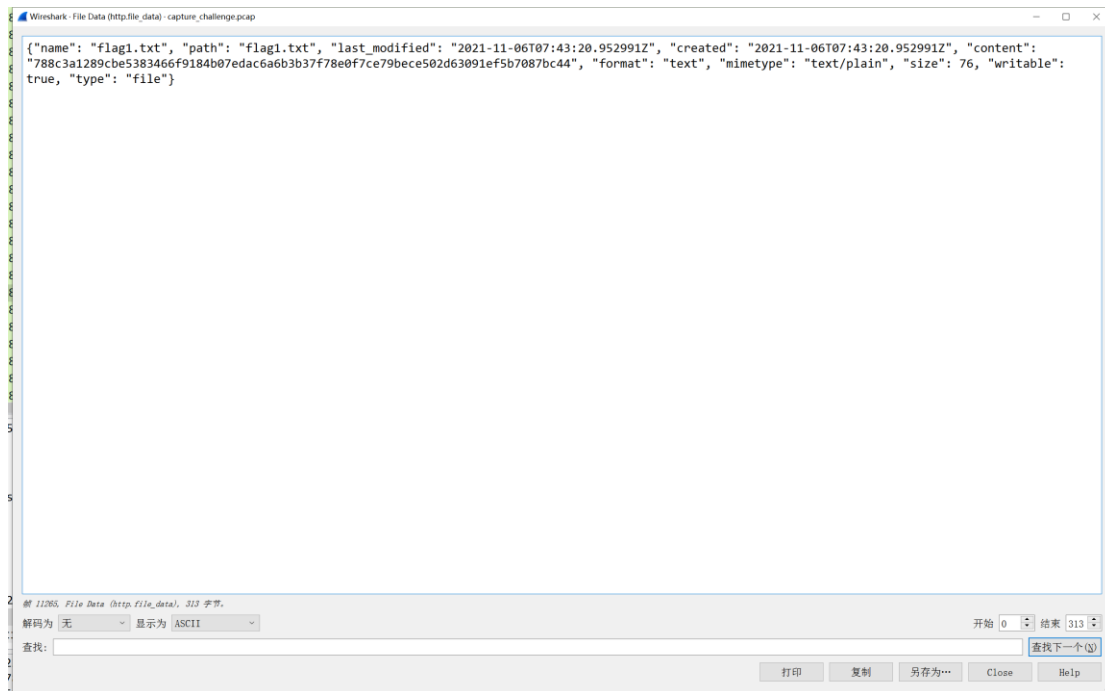
文件夹：car

做这道题也是我第一次用 WireShark，一开始看里面的流量记录出来 HTTP 协议的其他都看不懂。

先看 HTTP 协议的流量，发现应该是和 Jupiter Notebook 交互的记录，从中可以提取出一段代码，在 car/run.ipynb 里，大概就是怎么对 flag 进行异或运算加密的。

然后发现了 17s 的时候 GET 了 flag1 的文件，随后该文件以 json 返回，进行解密运算之后得到了 flag1。

11211 17.481109	192.168.17.128	192.168.17.1	HTTP/3_	411 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
11262 17.506183	192.168.17.1	192.168.17.128	HTTP	858 GET /api/contents/flag1.txt?type=file&format=text&_1636184605693 HTTP/1.1
11265 17.509723	192.168.17.128	192.168.17.1	HTTP/3_	770 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
11352 17.564007	192.168.17.1	192.168.17.128	HTTP	735 GET /custom/custom.js?v=20211106150620 HTTP/1.1
11353 17.564706	192.168.17.128	192.168.17.1	HTTP	431 HTTP/1.1 304 Not Modified



之后想得到 flag2, 发现 flag2 最后被以一个 7z 的方式从 HTTP 中返回了, 我下载了这个 7z, 发现要密码。

在 21s~70s 中是没有 HTTP 流量的, 所以应该是别的流量。

我发现大部分协议都是加密的, 只有很少的一些 web socket 有一些 Line-based text data, 这是我唯一看得懂的了, 仔细查看, 发现像是命令行的指令和回应, 于是从头到尾筛查来看, 发现这些指令做了一些事:

1. 安装了一个 python lsp 隐写库。
2. 使用这个隐写库将 flag2.txt 写到一个 wav 里, flag 长 76bytes
3. 将这个 wav 打包, 密码为: Wakarimasu! `date` `uname -nom` `nproc`

所以要恢复这个密码, uname 可以从之前的“mroot@you-kali-vm”找到, 系统应该就是 x86_64 GNU/Linux 吧, nproc 看后面有 cpu 信息, 但不知道这个虚拟机给了几核, 可以枚举, date 也是可以枚举的, 但 date 格式有多种, 我尝试了:

Sat Nov 06 15:44:}{ CST 2021

Nov 6, 2021 15:44:}{ 中国标准时间

枚举了很久, 最后还是没试出来, 我一度以为 nproc 或是 uname 是其他情况。

结果, 第二阶段公布提示后, 才知道竟然是 **12 小时制** 的, 最后枚举得出答案为:

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

叶子的新歌

文件夹: music

这道题高度好评, 花样非常多, 而且还有剧情!

由于受到 CTF-wiki 的启发, 以为三个 flag 都是音频隐写, 感觉不太熟悉, 太难了, 之前就没做这题。第二阶段才知道 metadata 中就有线索, 才后悔莫及。

Metadata 中有 aHR0cDovL2xhYi5tYXh4c29mdC5uZXQvY3RmL2xlZ2FjeS50Ynoy, base64 解码得到网站: <http://lab.maxxsoft.net/ctf/legacy.tbz2>

下载得到一个 tbz2 压缩文件, 一个 txt 提示另一个 img 是软盘镜像, 我直接按照网上的方法改名为 bin 打开了

然后是一个 MEMORY.zip 的压缩文件, 和一个密码提示“宾驭令诌怀驭榕喆艺艺宾庚艺怀喆晾令喆晾怀”, 这个密码从网上可以找到解码规则, 大概就是在数字和汉字之间做了一个映射。程序在 music/run.ipynb。密码为: 72364209117514983984

打开之后是一个 left.bin 和一个 right.bin, 还有提示: “找不同”, “NES”, 用 010 Editor 打开两个 bin 发现只有一点点不同, 写了个程序将不同的字符提出(run.ipynb)。发现文件开头是 NES, 我下载了一个超级玛丽的镜像, 开头也是 NES, 感觉这就是 NES 镜像了。

但放到 Fceux 模拟器里打不开, 应该是找不同处理错了, 一开始匹配时我只想前看来一位, 结果出现了很多长度为 2 的不同字符, 之后我改为多看几位, 每次不同的长度就是 1 了, 现在就能打开了。

没想到竟然也是一个超级玛丽的游戏, [一开始我还跟随题目沉浸在怀旧的氛围中, 决定手动通关, 只开一个无线生命的外挂](#), 但我实在是太菜了, 恼羞成怒之后开了一大堆秘籍, 跳关到 8 完成了游戏。

然后屏幕上出现了一个网址, 这个网站需要填入一个密码, 我试了之前的汉字密码发现不对, 试了 metadata 也不对。

最后我觉得试着按照提示的“虚拟化工具”正式地打开一下软盘, 用 VirtualBox 加载之后果然出现了密码, 得到了 flag3。同时软盘直接给出了 flag2。

其实很很想多试试解出 flag1, 感觉 flag1 应该是真正和音频有关的, 但是研究生狗事情太多, 没有时间了。

在线解压网站

文件夹: unzip

发现服务器代码直接做的解压, 没有什么安全措施。我从网上看了怎么把一个软连接压缩起来, 好像用—symlink 选项就行, 然后压了一个指向根目录的软链接, 看到了 flag。

然后又压了指向 flag 的软连接, 得到了 flag。

Flag 即服务

想通过 ./ 看看能不能拿到别的文件, 发现能拿到 package.json。在 <https://prob11-flrde9t3.geekgame.pku.edu.cn/api/..%2fpackage.json> 拿到了 package.json, 里面有后端源码的链接, 然后 index.js 里有关于 flag1 的信息:

```
if(FLAG0!=='flag${0.1+0.2}')  
  return;
```

直接在浏览器 console 里对 flag\${0.1+0.2} 进行求值, 得到 flag1。

最强大脑

看了题学习了一下 brainfuck，先试了一个网上的 hello world，运行正常。
我又试了一下走一步输出一下当前位置，竟然直接得到了一个 flag!

密码学实践

文件夹：passwd

看了源码，发现第一个 flag 和一句话都是用的 PublicKey 做的加密，查看加密操作只在 4x8 个 bits 内进行，主要是异或，模拟了一下，发现最后等价于每个数异或上一堆 Key，数的主体是 c, d, a^c, b^d 。

这样的话，完全可以通过明文和密文推断出异或那的一团东西，然后进行逆运算，就可以解密了，顺利解出 flag1。

第二问主要是通过模数的长度不够，让我们可以构造与目标内容同余的数，然后利用长度相减得到负数，利用 python 的负下标特性，使得最终 dec 得到的 name 为 Alice。

同时可以让 info 和 P 同余，这样最终用来加密的 NewKey 还是 PublicKey。