Week1-Nse4u WP

MISC

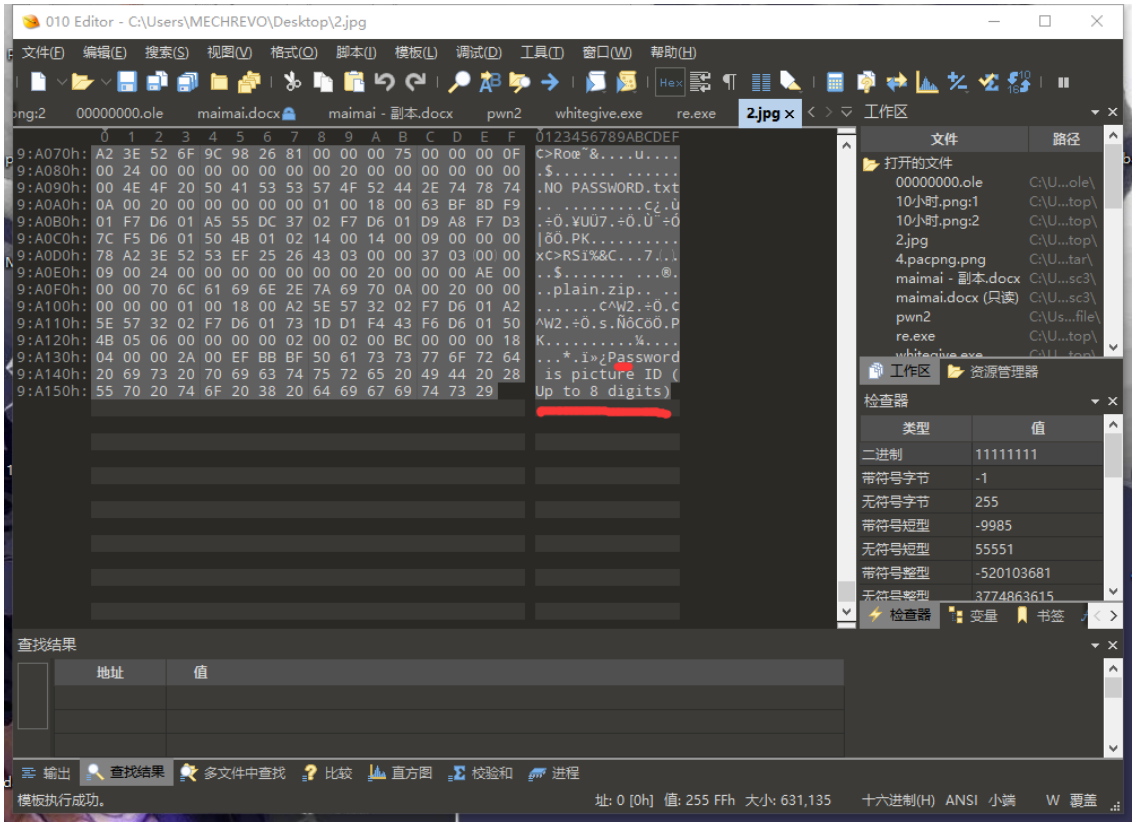
1.base 全家福

1. 打开base在线解密网站： https://www.qtool.net/baseencode
2. 先base64解码

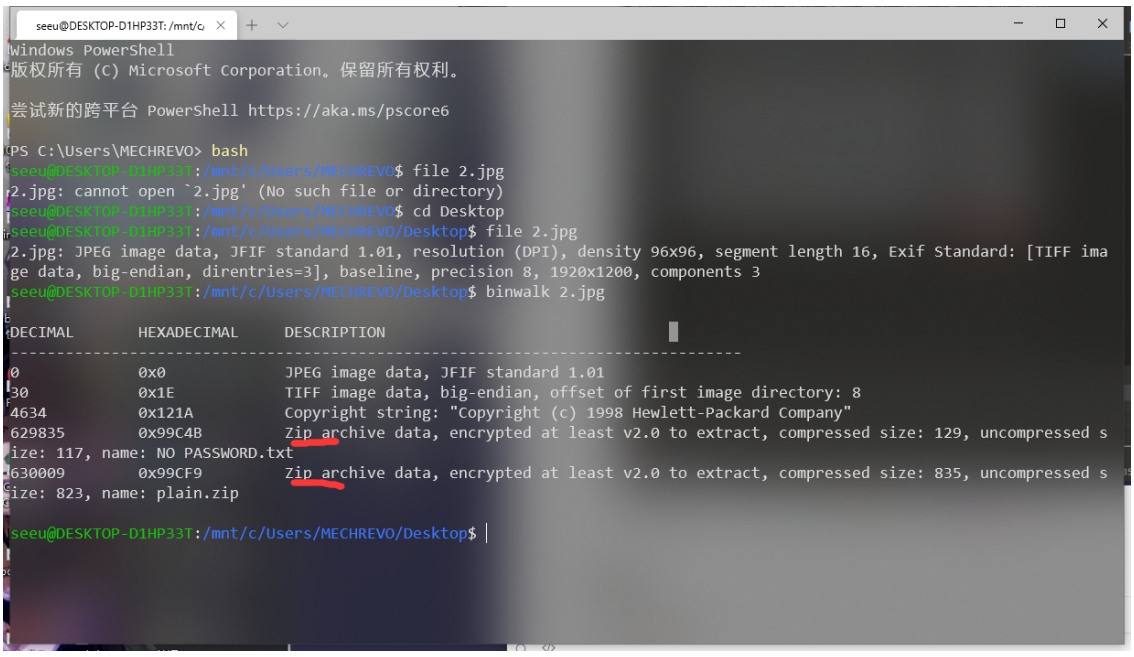
3. 然后base32
4. 然后base16 得到 hgame{We1c0me_t0_HG4M3_2021}

（因为采用一种base编码的方式的密文用其他base解码方式是解不了的，网站上也不会有返回值，所以可以试一下就就可以得出有没有采用正确的base解码，所幸这题的解码次数比较少，不用写个程序跑）

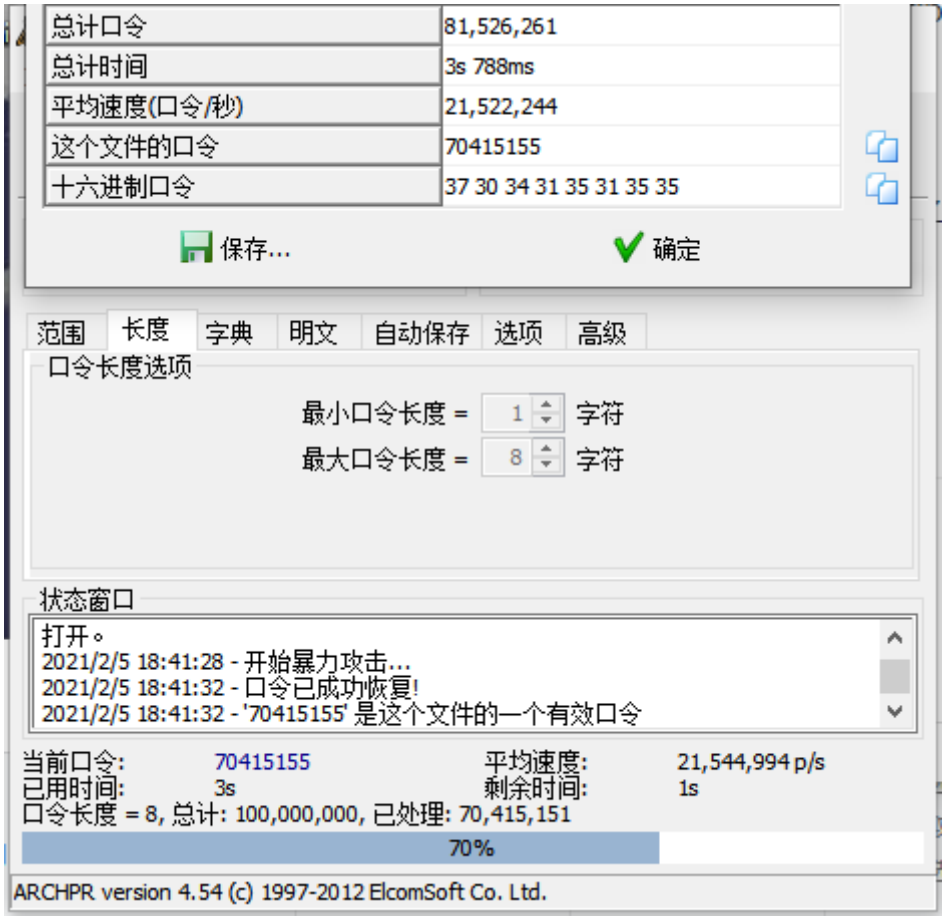
2.不起眼的压缩包养成方法

1. 首先打开网页后出现一张图片，右键另存为保存
2. 根据提示暗藏玄机，于是使用010editor打开看看
3. 

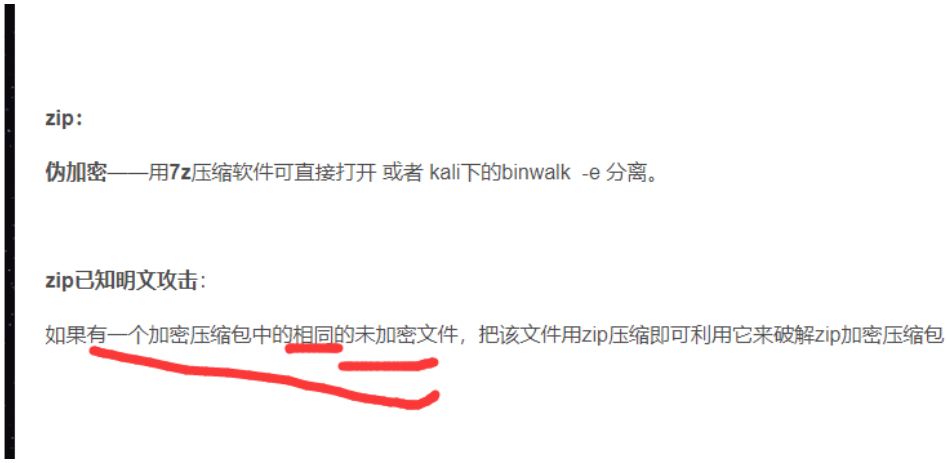
4. 发现末尾有个password 但是没有遇到打开密码，于是binwalk一下试试
5. 发现里面藏了个zip



6. foremost 分离一下,得到jpg和zip，zip打开需要密码
7. 百度如何破解加密zip----得到软件 :Arghpr.exe
8. 根据提示password up 8 digits,爆破之



9. 得到打开密码
10. 打开后得到一个压缩包，和一个nopassword.txt，有密码，爆破不开，百度ctf常见zip加密方法



zip：

伪加密——用**7z**压缩软件可直接打开 或者 kali下的binwalk -e 分离。

zip已知明文攻击：

如果有一个加密压缩包中的相同的未加密文件，把该文件用zip压缩即可利用它来破解zip加密压缩包

11. 正好解压出来的nopassword.txt在要打开的压缩包中也有一个nopassword.txt
12. 于是尝试zip明文攻击
13. 打开后还有一个压缩包，根据nopassword.txt中没有密码的提示，尝试为伪加密

压缩源文件数据区：

50 4B 03 04：这是头文件标记（0x04034b50）

14 00：解压文件所需 pkware 版本

00 00：全局方式位标记（有无加密） 头文件标记后

08 00：压缩方式

5A 7E：最后修改文件时间

F7 46：最后修改文件日期

16 B5 80 14：CRC-32校验（1480B516）

19 00 00 00：压缩后尺寸（25）

17 00 00 00：未压缩尺寸（23）

07 00：文件名长度

00 00：扩展记录长度

6B65792E7478740BCECC750E71ABCE48CDC9C

压缩源文件目录区：

50 4B 01 02：目录中文件文件头标记(0x02014b50)

3F 00：压缩使用的 pkware 版本

14 00：解压文件所需 pkware 版本

00 00：全局方式位标记（有无加密，伪加密的关键）

08 00：压缩方式

5A 7E：最后修改文件时间

F7 46：最后修改文件日期

16 B5 80 14：CRC-32校验（1480B516）

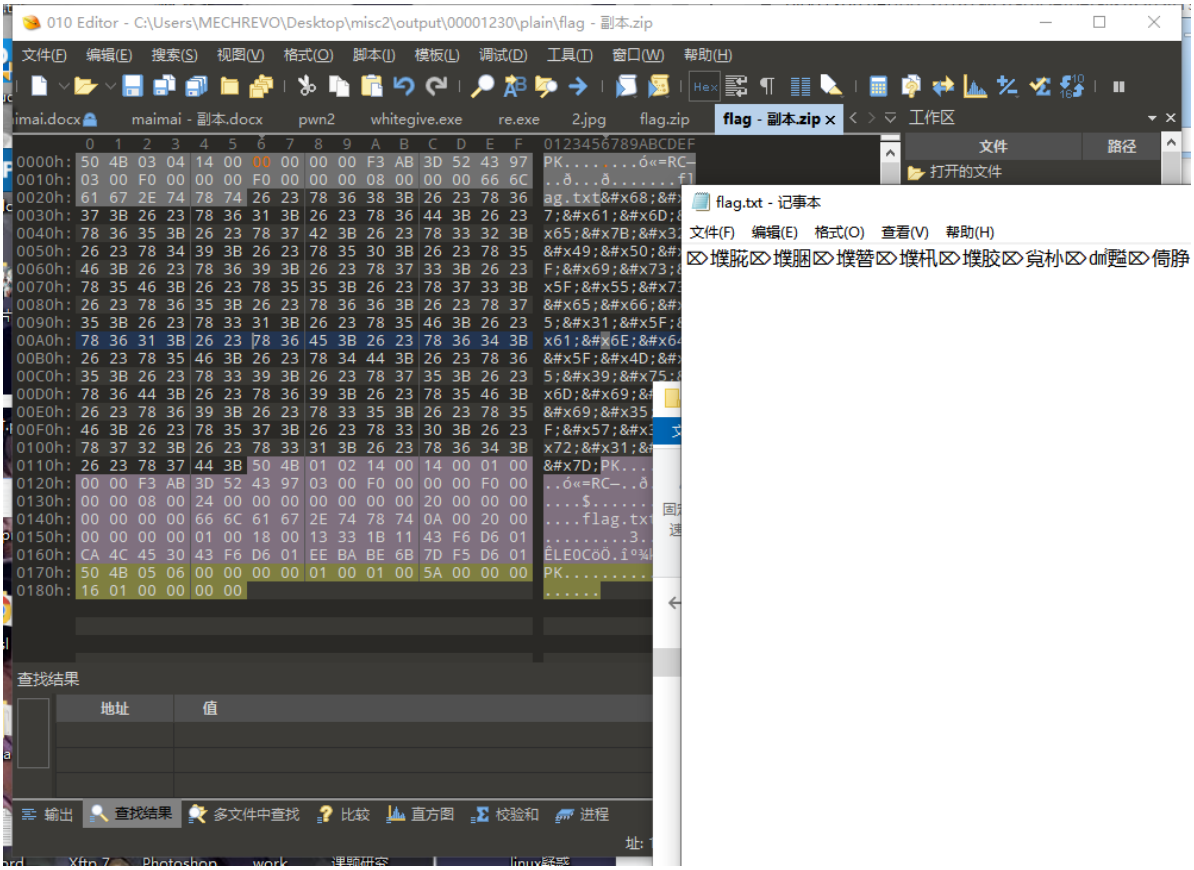
19 00 00 00：压缩后尺寸（25）

17 00 00 00：未压缩尺寸（23）

07 00：文件名长度

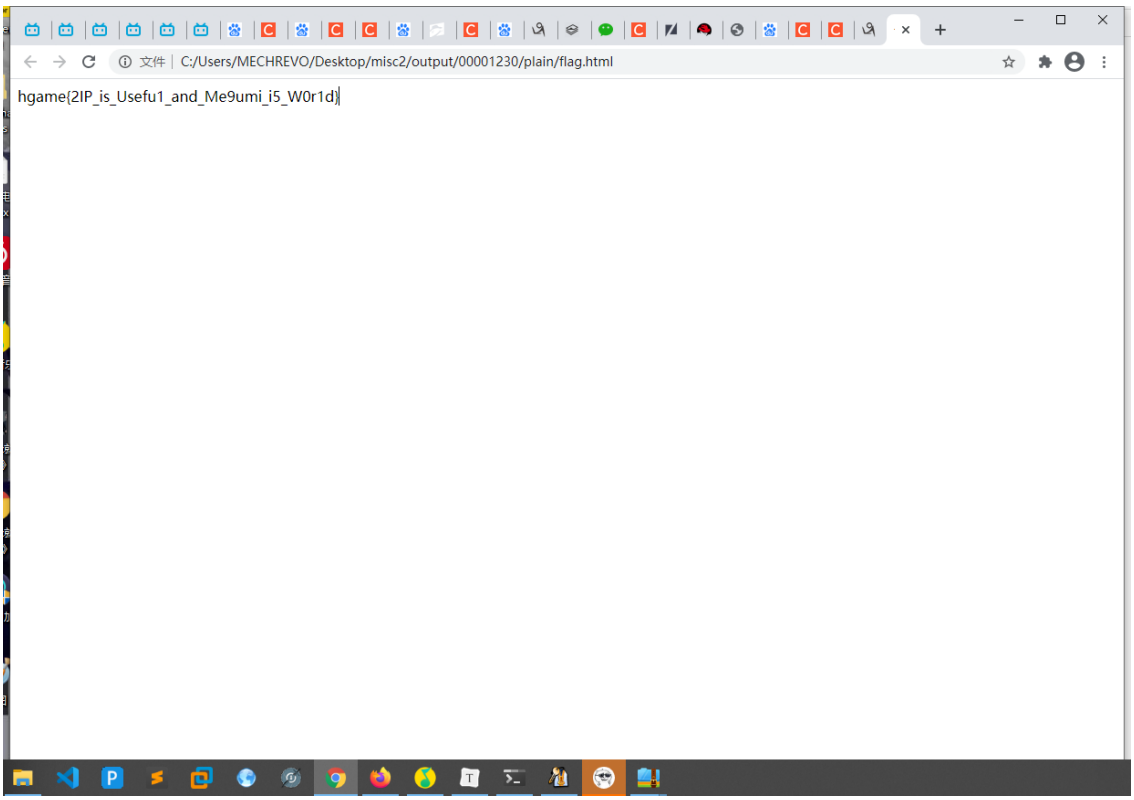
24 00：扩展字段长度

00 00：文件注释长度



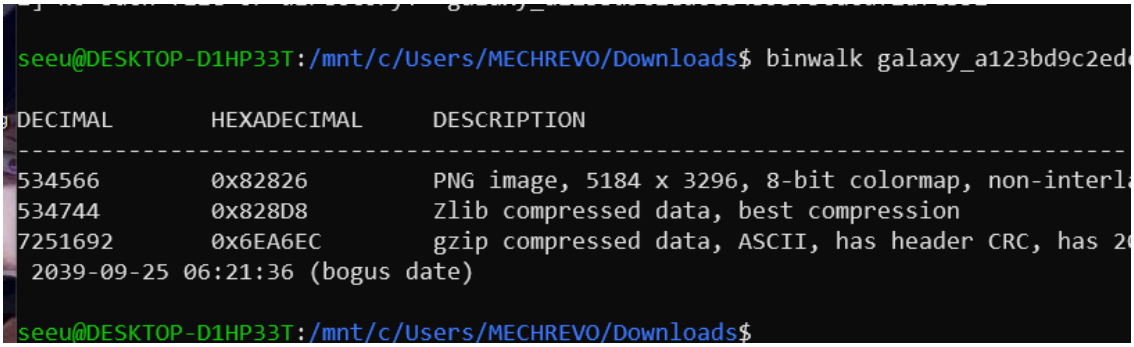
14. 修改压缩包数据区即可打开

15. 根据Akira学长的提示，用记事本打开是会出现乱码的问题，要用浏览器打开（这里我猜是编码方式不同的原因），然后用浏览器打开，flag出现！



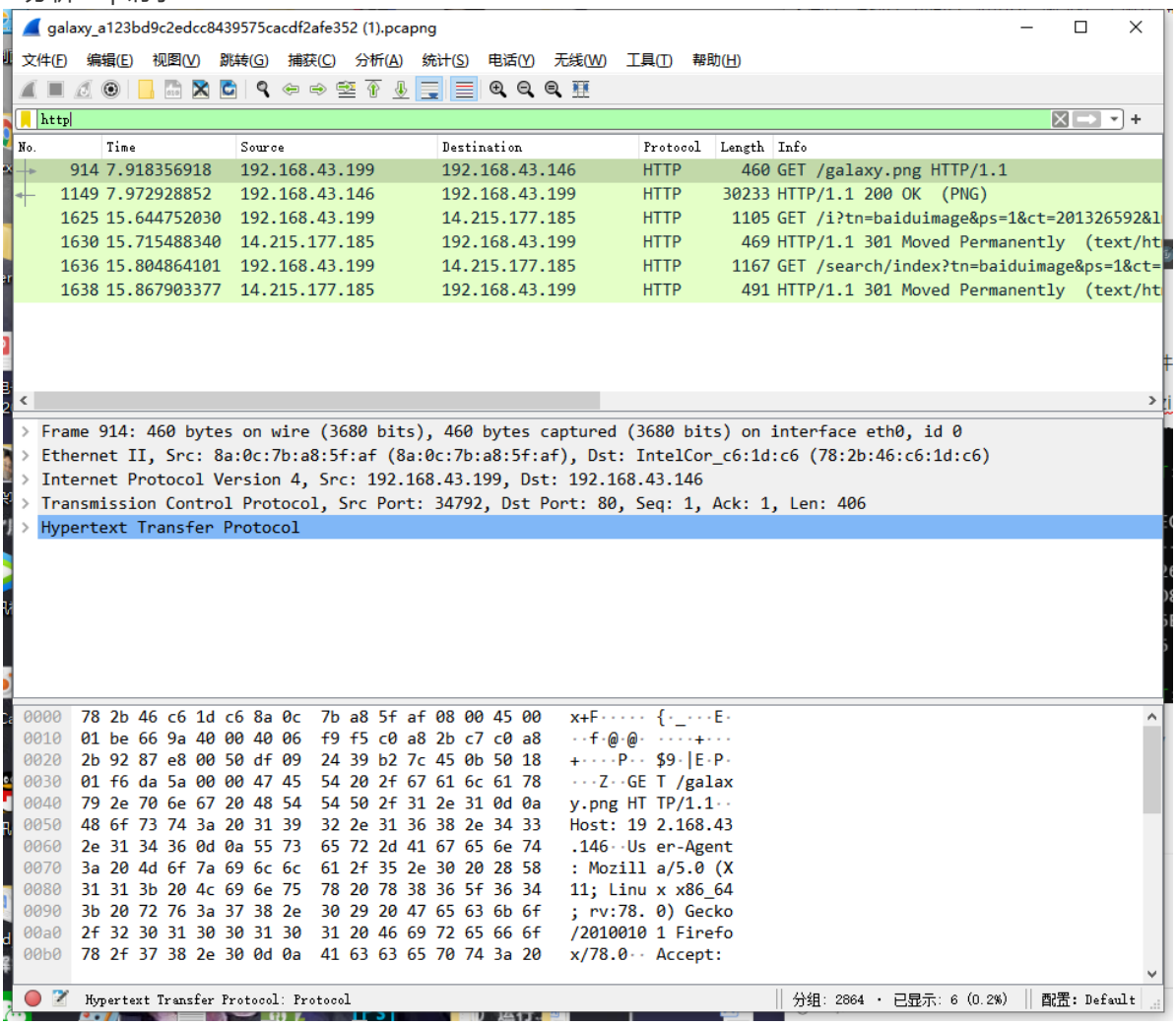
3.Galaxy

1. 下载后得到一个.pcapng文件，有了上一题文件里藏文件的经验，先binwalk一下
2. 发现里面有个png、zlib、gzip，但是foremost分离不出来（dd试了很久，也分不出，可能是我菜的原因）



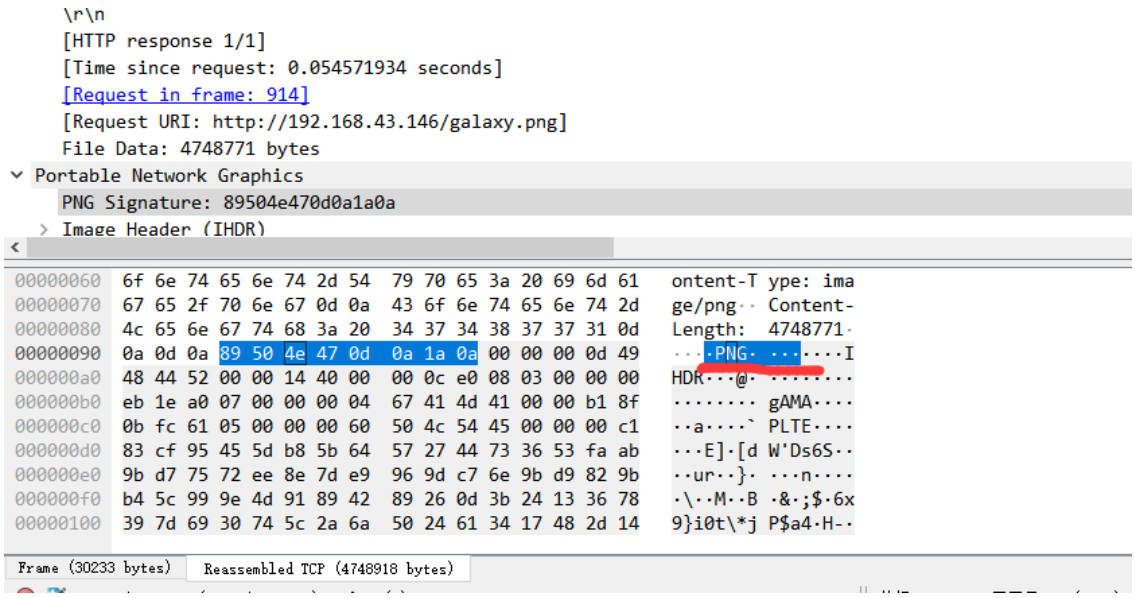
3.于是百度，得知这是个流量包，要下载wireshark，于是下载后打开流量包分析

4.分析http请求



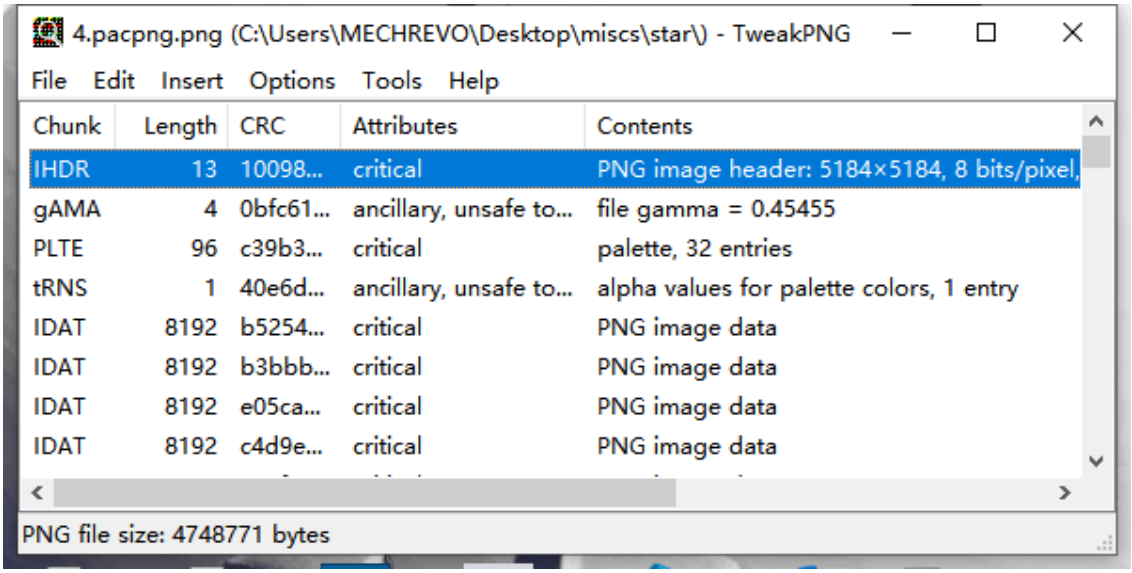
5. 发现有个特别大的请求

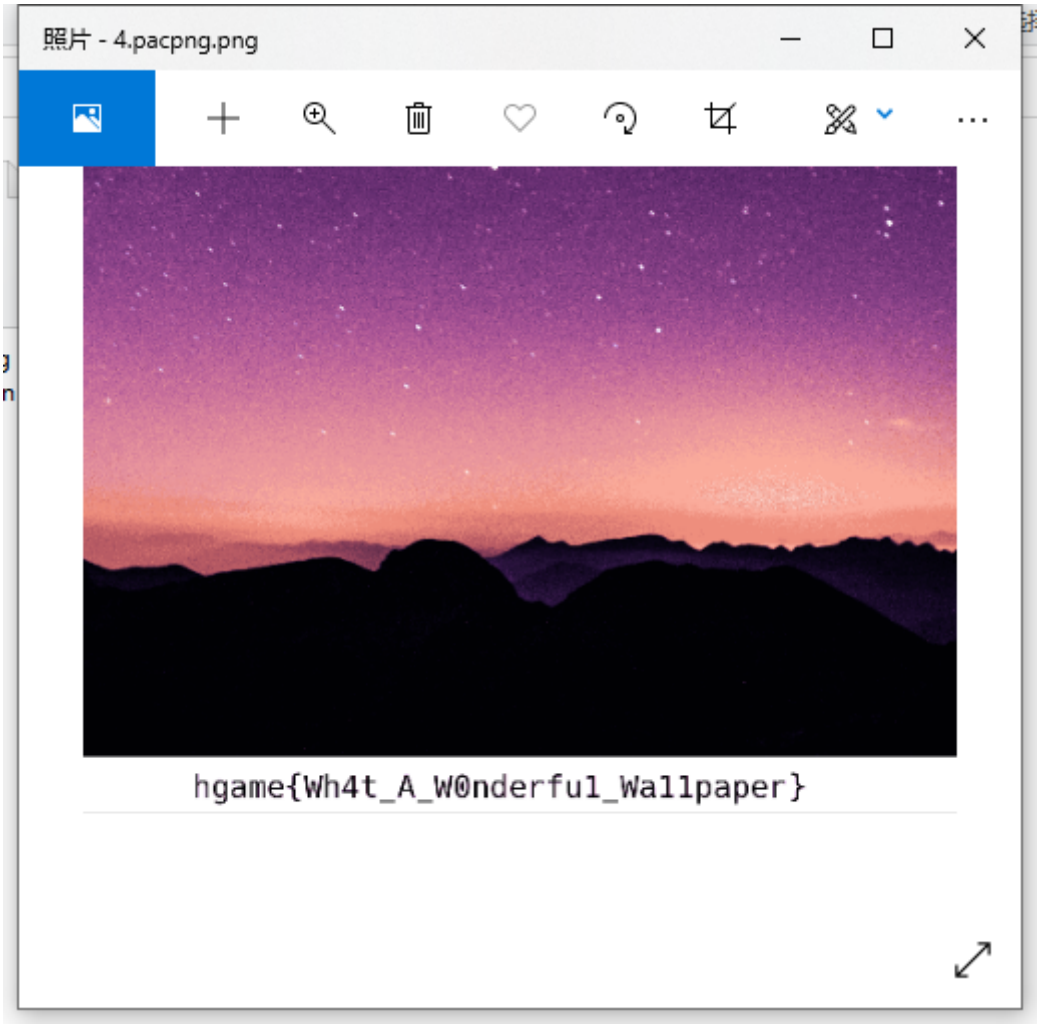
6. 打开发现里面有个png头



7. 根据题目提示，信物是星空图片，于是导出看一看

8. 打开属性一看 高度和宽度不一致，正好nepnep的直播课上有讲到，猜想这可能把隐藏在下面了，于是打开tweakPNG修改高度，flag出现！





9.然后因为太菜了，提交flag的时候还输错了好几次

4.Word RE:MASTER

1. 下载后解压得到两个doc文档，一个叫first,一个叫maimai，

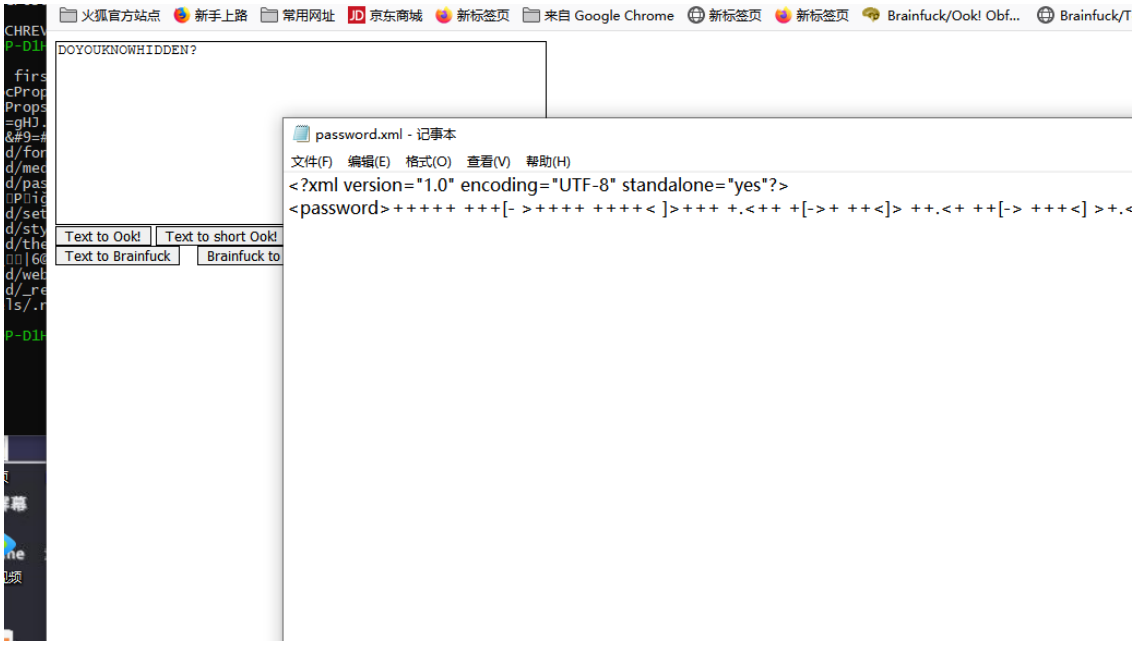
2. first打开是一个奇怪的图片和一段文字

Fuck！我的脑子好疼！这可能是音游瘾发作最严重的一次,躺在床上很想打交互，嘴里念叨：O-oooooooooo AAAAE-A-A-I-A-U JO-ooooooooooooo AAE-O-A-A-U-U-A E-eee-ee-eee AAAAE-A-E-I-E-A JO-ooo-oo-oo-oo EEEEO-A-AAA-AAAA,不行我得在brainpower 耗尽前把密码记下来。

3. 用010editor打开maimai，在底部发现

```
.................
Password is IN t
he other docx...
 IN the other do
cx..word IN the
other docx..+++<
] >+.<+..++[-> -
--<] >-.++ ++++.
 <+++[ ->--- <]>
-. +++.+ .++++ +
+++. <+++[ ->---
..<]>-- ----. +.
--- --..+ .++++
+++++ .<+++ [->-
- -<]>- ----- .<
............brai
n..........fuck
```

4. 于是猜测密码要用另一个打开，顺便百度brainfuck
5. 百度得知ook加密和brainfuck加密，maimai最后一段，符合brainfuck的编码方式，于是解密，解除来啥也没有
6. 于是根据提示，密码在另一个文档【里面】，于是foremost first.docx
7. 得到一个jpg和一个zip ，解压zip，在里面发现很多文档，其中有一个叫password.xml
8. 打开password.xml，发现是brainfuck编码
9. https://tool.bugku.com/brainfuck/
10. 在线解码得到密码：DOYOUKNOWHIDDEN?

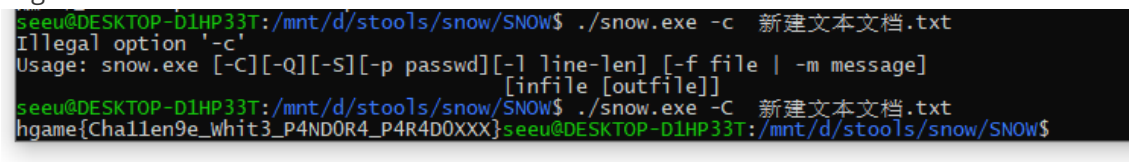


11.打开后发现是奇怪的图片和奇怪的文字

12.百度ctf doc文档 misc，得知doc隐写术中有一种隐藏文字隐写，勾选打开，发现奇怪符号

13. 百度ctf 加密方式，得知有一种加密叫snow加密，正好符合这里的样子，于是下载snow.exe
14. 但是打不开，（尝试把doc后缀改为txt，更打不开了）

15. 根据Akira学长指点，要取消文字隐藏，才能复制出来，于是复制出来到txt文本中进行解密，得到flag

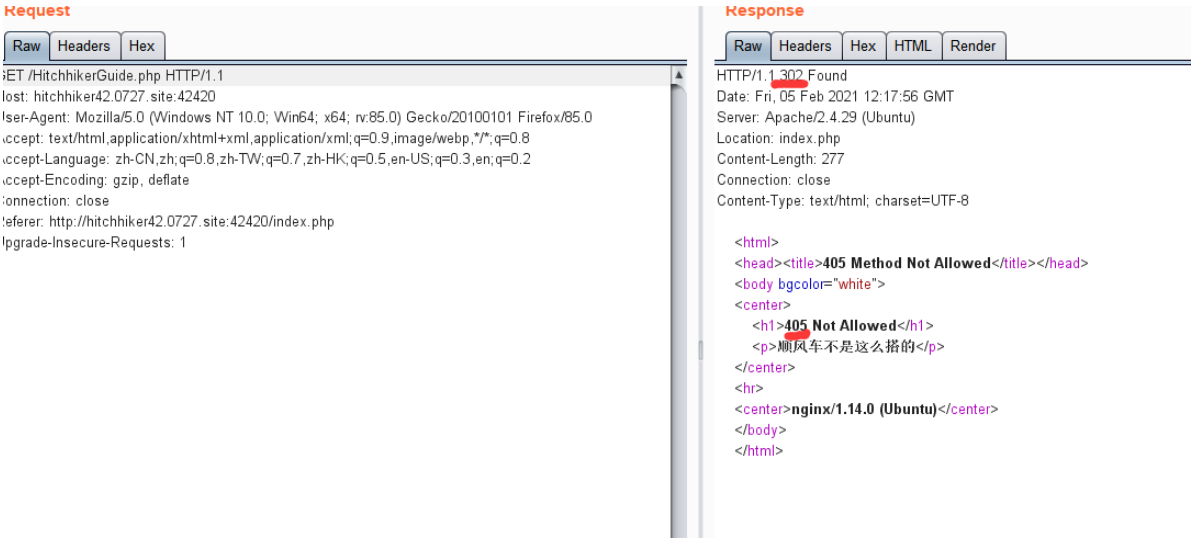


web

1. Hitching_in_the_Galaxy

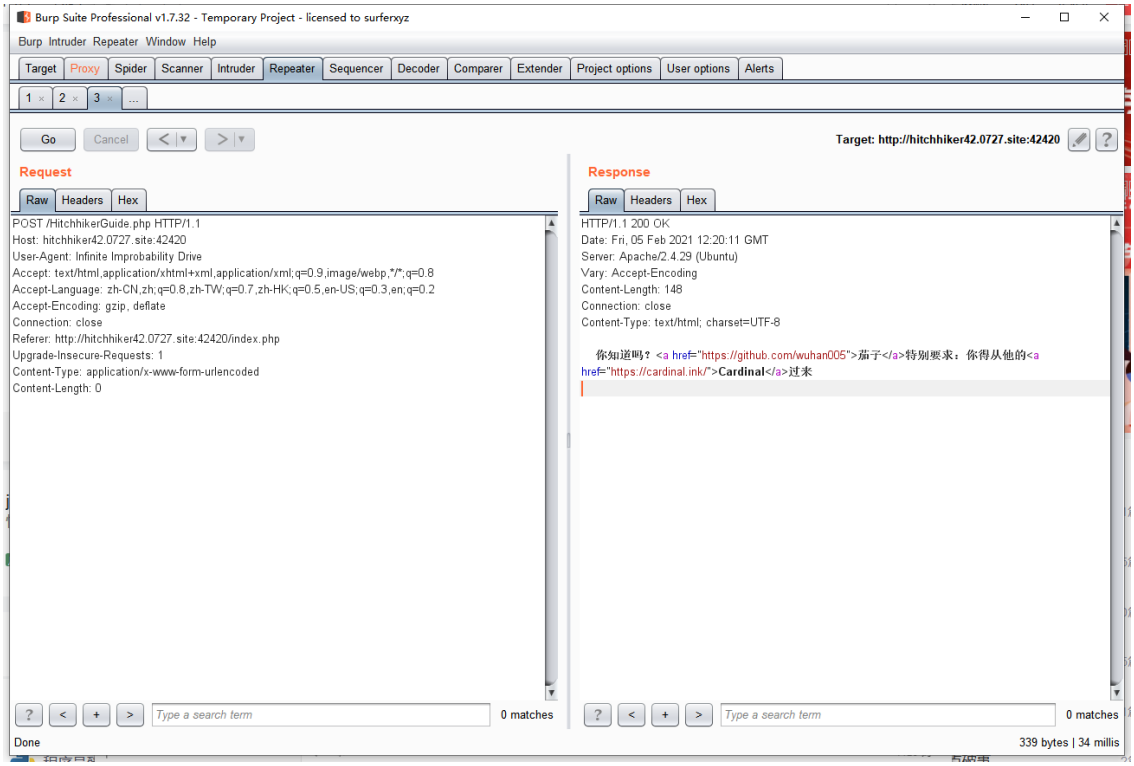
1. 打开发现404，还有一个超链接

2. 点进去超链接，发现有回到了404，应该是被302了,于是burp抓包看一下

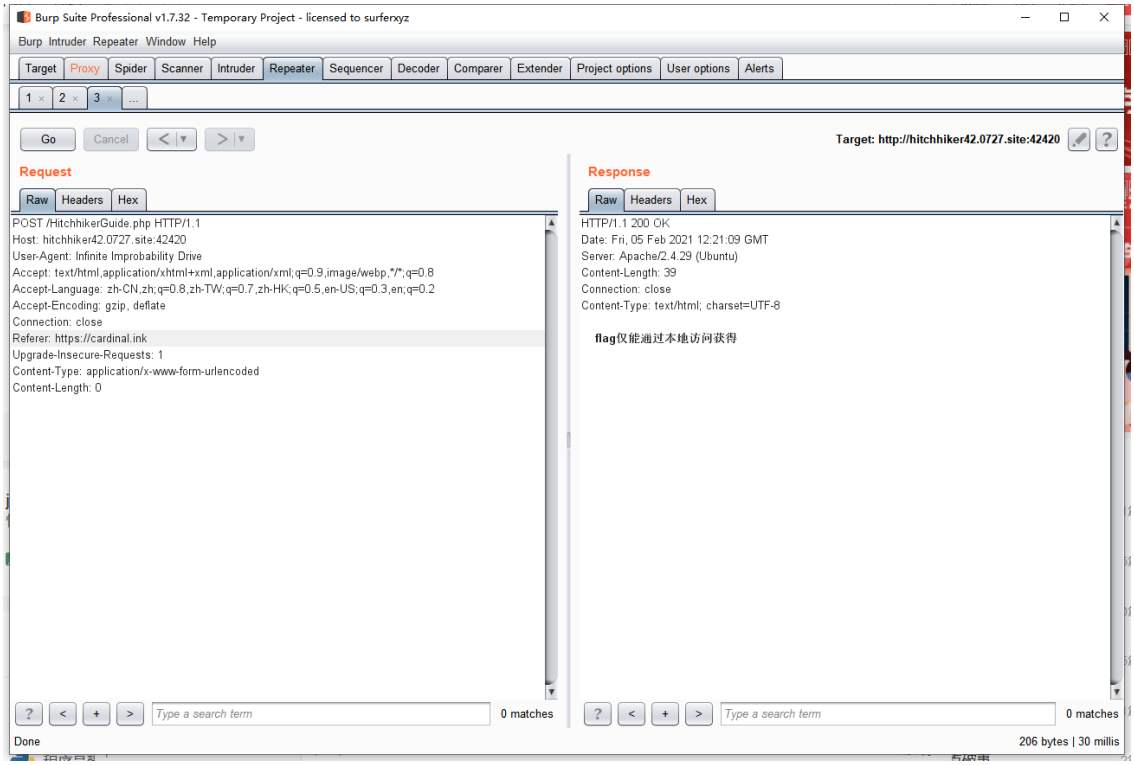


3. 发现一个 302 和一个405 ， 说顺风车不是这么搭的，于是change requestmethod

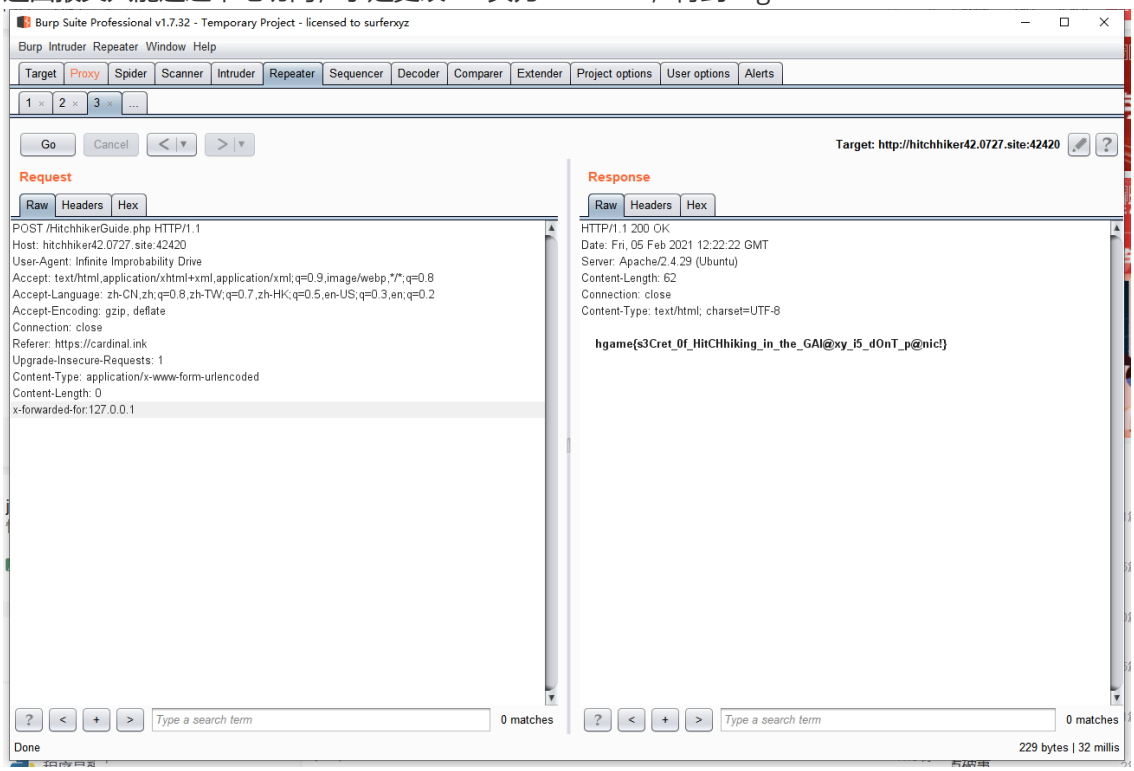
4. post之后发现说 只有【无限非概率引擎】可以到达，那就是用户不对咯?



5. 然后给了个提示，只有从他的链接过来才行，复制给的href,更改referer头

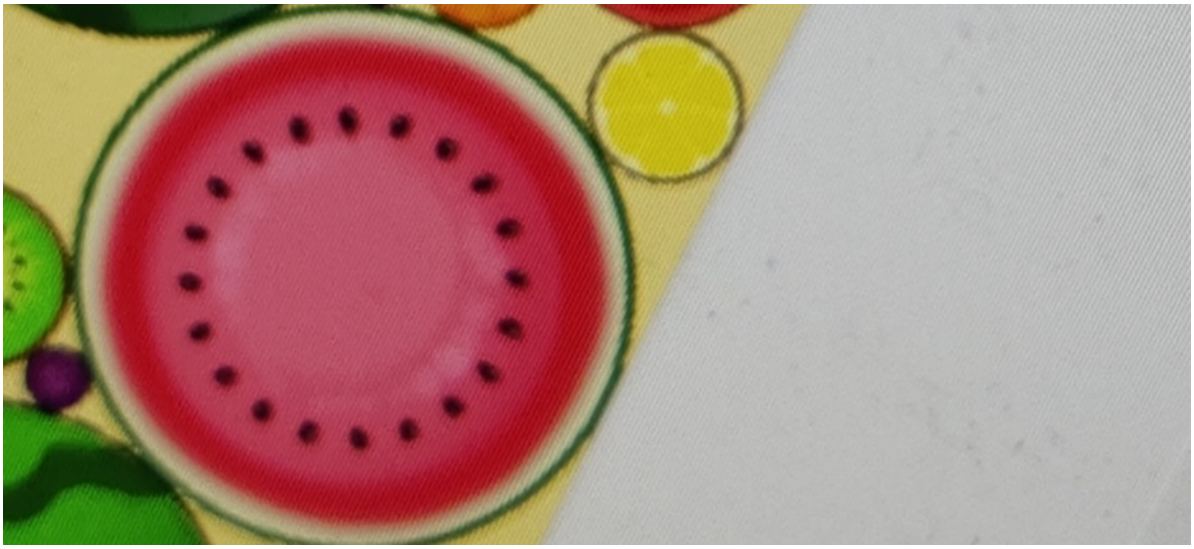


6. 返回报文只能通过本地访问，于是更改XFF头为localhost，得到flag

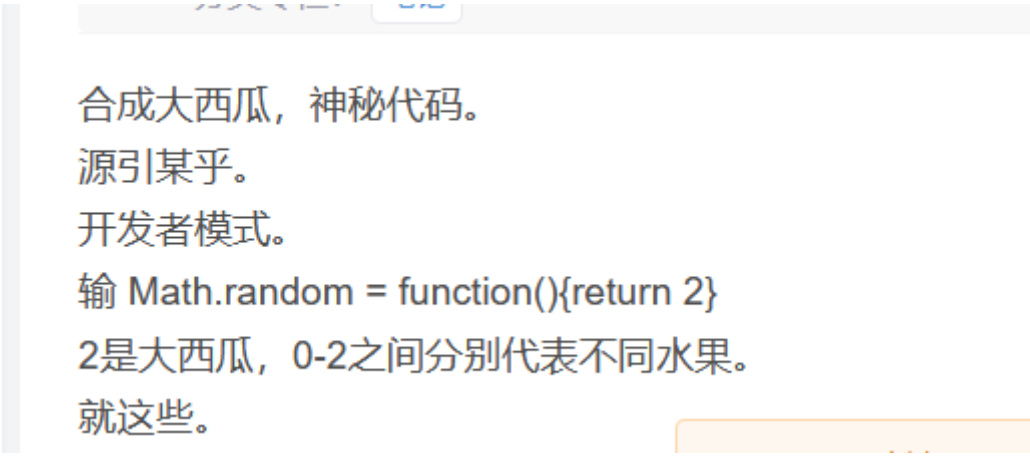


2. watermelon

2024
游戏结束



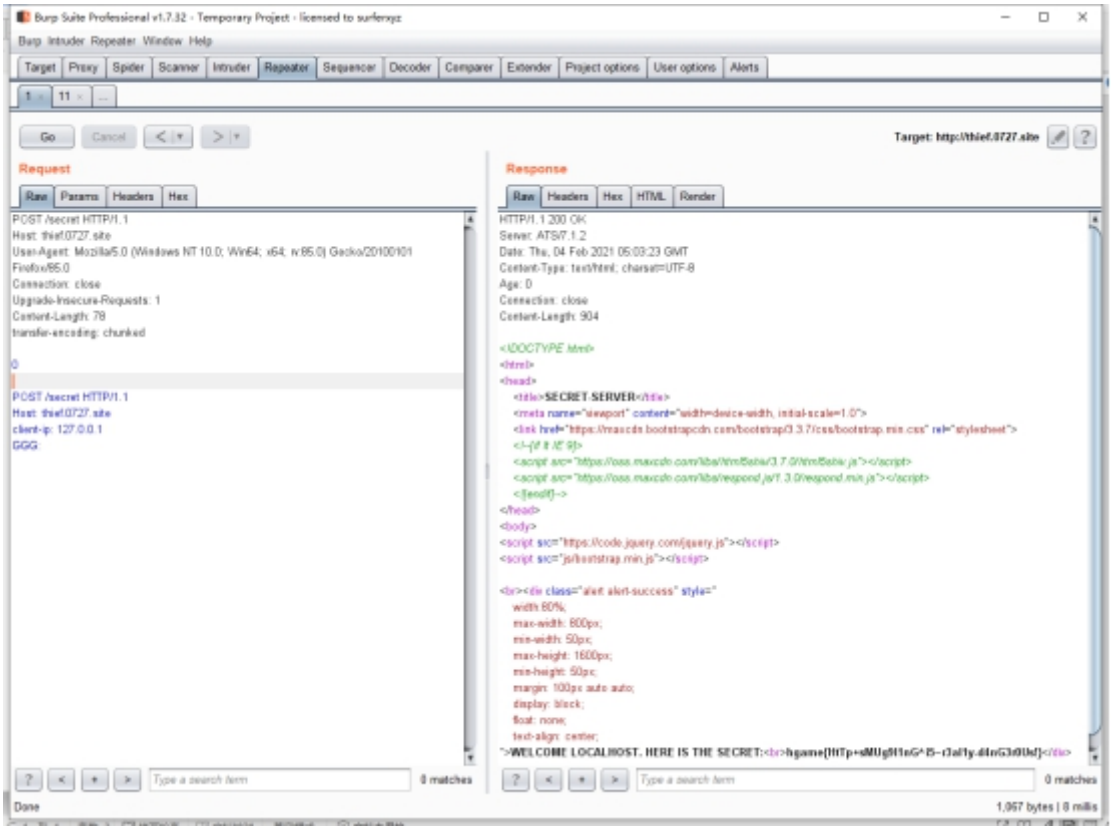
1. 点击去发现是一个玩大西瓜的网页，先F12看看源码
2. 代码太长了，变量名记都记不住尝试看看抓包能不能修改分数
3. 发现似乎没有数据包发送出去
4. 于是百度大西瓜源码，试试看能不能找个有注释的代码看看
5. 在CSDN上发现神秘代码，输入之，起效！



6. 于是尝试输入
7. 控制台打开，修改 Math.random = function(){ return 1.0}
8. 于是可以每次都掉大番茄，点点点就到了2000分

3. 宝藏走私者

1. 一开始看到localhost才能登录，想到的是改xff头，发现不行，有验证机制
2. 根据资料里的内容，发现这里是ats7.1.2反向代理
3. 根据资料里请求走私漏洞，对这里进行测试
4. 发现在发送cl:0 te: 100 时，请求超时，判断这是cl-te的前后端结构
5. 于是进行请求走私攻击
6. 使cl为正常长度 ，te为0，于是剩下的信息就会留在缓冲区里，等到下一个信息到来时，添加在下一个信息的头部
7. 由于已经到达后端服务器缓冲区，被后端服务器认为已经通过了前端的安全验证，在缓冲区里添加的client-ip被认为时合法的请求
8. 这个合法的请求添加在下一个头部，返回到我的电脑来，得到flag



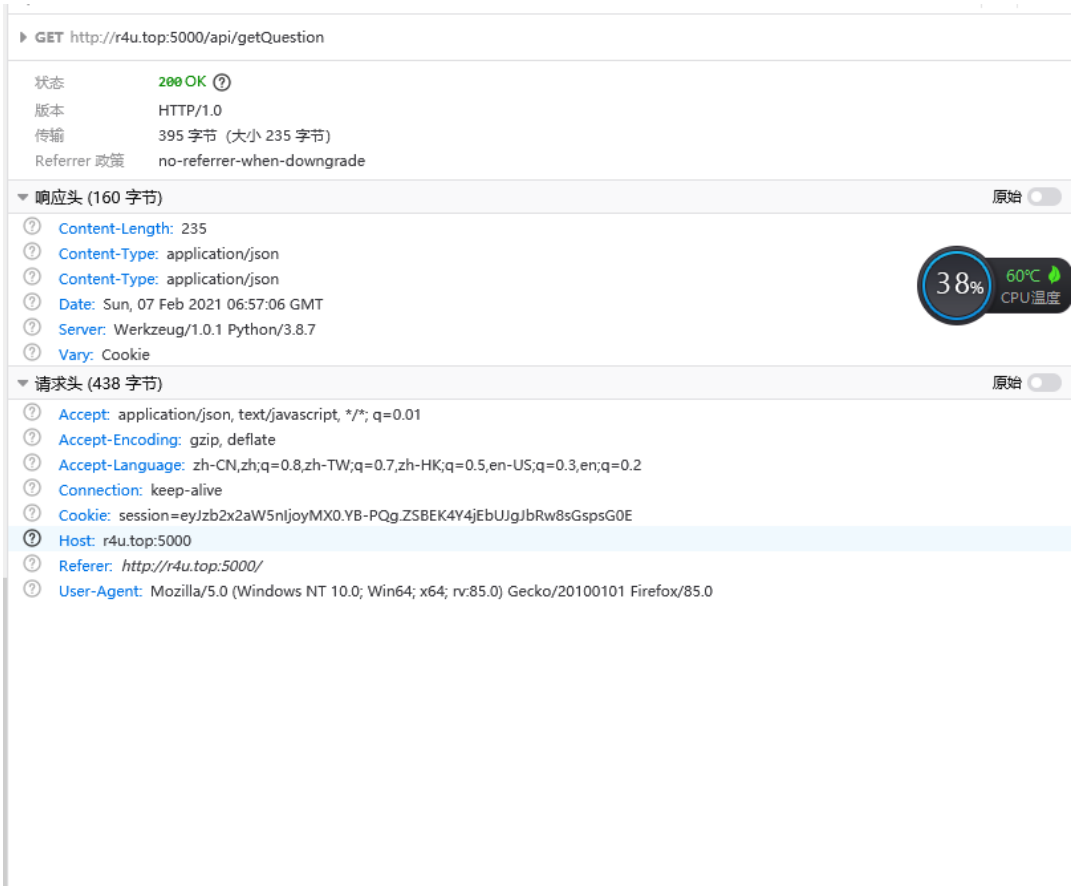
4. 智商检测机

1. 登录之后发现是一个页面，要做高数题
2. 先抓包看看有没有可以直接修改分数的
3. 发现没有，于是看看提交的网络请求过程
4. 发现是一个ajaix请求
5. 一次发送答案（正确的话），会返回3个响应
6.

200	P...	r4u.to...	verify	jquery-3....	json	269 字节	16...
200	GET	r4u.to...	getQuestion	jquery-3....	json	396 字节	23...
200	GET	r4u.to...	getStatus	jquery-3....	json	174 字节	15...

7. 分别是verify，这个是判断答案的接口
8. getQuestion，这个是获取下一题答案的接口
9. getstatus这个返回当前做了几题
10. 于是考虑可以写个python爬虫，用以反复提交

1. 先确认获取问题的页面



，是这个接口

1. 然后想如何获取问题，发现每次提交正确会有一个Set-cookie,于是猜测是靠cookie判定当前坐到了哪一题
2. 获取cookie应该在post了正确答案之后
3. 再post下一个

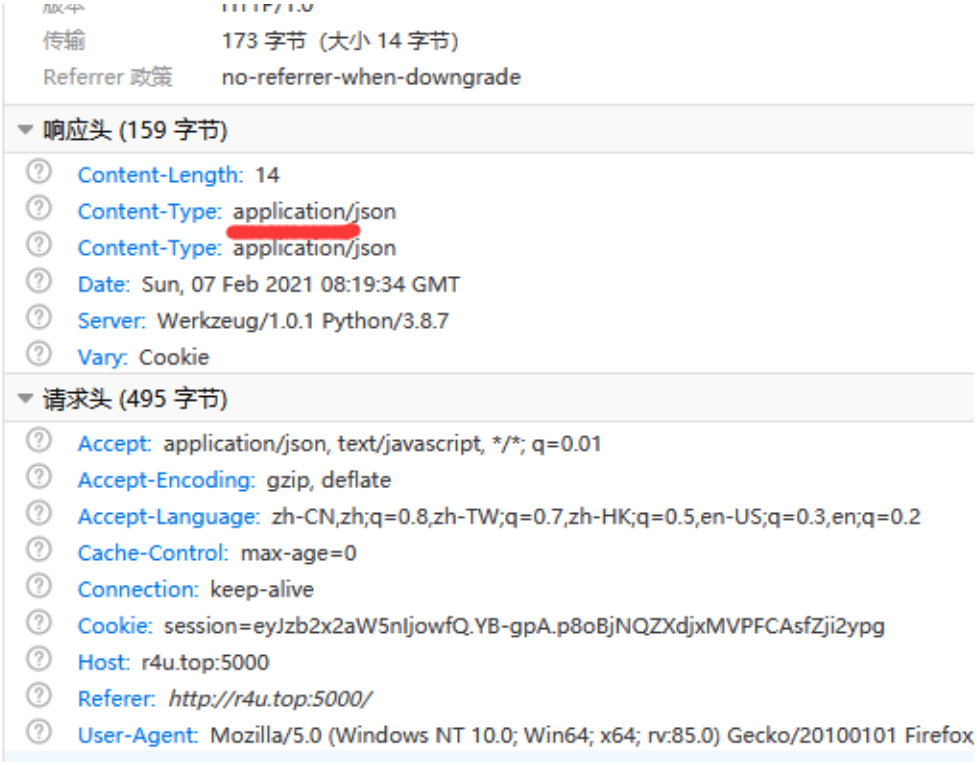
11. 查看检测机页面源码，发现有一个js接口，点进去发现

12.

```
function getStatus(){
    $.ajax({
        type:"GET",
        url: "/api/getStatus",
        dataType:"json",
        success:function(data){
            let solving = data['solving']
            $("#status").text(solving);
            if(solving === 100)
                getFlag();
        }
    });
}

function getQuestion(){
    $.ajax({
        type: "GET",
        url: "/api/getQuestion",
        dataType: "json",
        xhrFields: {
            withCredentials: true
        },
        crossDomain: true,
        success:function(data){
            $('#integral').html(data['question']);
        }
    });
}

function getFlag(){
    $.ajax({
        type: "GET",
        url: "/api/getFlag",
        dataType: "json",
        success:function(data){
            $('#flag').html(data['flag']);
        }
    });
}

function init(){
    getQuestion();
    getStatus();
}

function submit(){
    $.ajax({
```

13. 发现有个getflag接口，该接口就是做完100题之后要擦混入的cookie接口，于是得知之后，开始调试代码

14.



15. 查看返回头得知，这里只收json数据，于是要改answer加一个json.dumps()
16. 此时已经想好解体步骤
    1. 获取问题
        1. 爬取页面

{"question":"<math><mrow><msubsup><mo>\u222b</mo><mrow><mo>-</mo><mn>75</mn></mrow><mrow><mn>38</mn></mrow></msubsup><mo>(</mo><mn>5</mn><mi>x</mi><mo>+</mo><mn>16</mn><mo>)</mo><mtext><mi>d</mi></mtext><mi>x</mi><mtd/></mrow></math>"}

        2. 利用正则表达式得到要算的数字

```python
def getQuestion():
    response = requests.get(url=geturl,headers=headers)
    res = response.text
    print(res)
    p = re.compile(r'([\d]{1,5})<')
    p1 = re.compile(r'(-).*?([\d]{1,5})<')
    p_res = p.findall(res)
    p1_res = p1.findall(res)
    #print(p_res)
    #print(p1_res)
    for i in range(0,len(p1_res)):
        for j in range(0,4):
            if(p_res[j])==p1_res[i][1]:
                p_res[j]='-'+p_res[j]
                p1_res[i]=('','9999')
    run = [float(x) for x in p_res]
    print(run)
    return run#返回列表对象，列表对象里面是取得的几个问题数字

```

    2. 解决问题，就是写一个计算答案的函数就行了

```python
def counts(t,x,c):#用于计算
    return 0.5*x*t**2+c*t
```

```
def solveitone(question):
    down = question[0]#积分上限
    up = question[1]#积分下限
    x = question[2]#一次项
    c= question[3]#常数项
    a = counts(up,x,c)#计算积分上限值
    print(a)
    b = counts(down,x,c)#计算积分下限值
    print(b)
    answer = a-b#减一减得出定积分答案
    print(answer)
    return answer #返回一个数字，是要post的答案
```

3. 更新answer，然后post回去

```
def postAnswer(data):
    response =
requests.post(url=posturl,data=data,headers=headers)
    print(response.text)
    head = response.headers
    x = requests.structures.CaseInsensitiveDict(head)
    js = dict(x)
    session = js['Set-Cookie']
    return session#返回一个字符串对象，是Set-Cookie的值，用于判断题
目第几题
    #就是把答案post过去，然后可以更新cookie的值，然后就可以得到下一题
的题目信息
```

1. 需要注意的是，传入的data必须要json.dumps

4. post之后得到新的cookie，更新cookie

```
data['answer'] = answer
session = postAnswer(json.dumps(data))
```

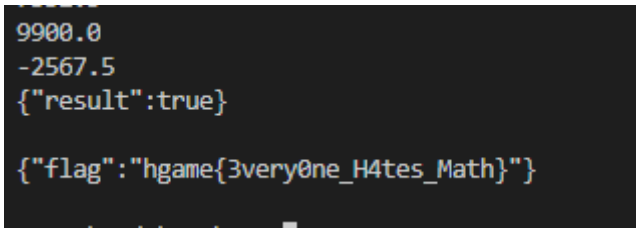
5. 重复99次

```
i=0
for i in range(0,100):
    headers['Cookie'] = session
    question = getQuestion()
    answer = solveitone(question)
    data['answer'] = answer
    session = postAnswer(json.dumps(data))
```

6. 带着最后的cookie去访问/api/getFlag页面，获取flag

```
headers['Cookie'] = session
response = requests.get(url=url2,headers=headers)
print(response.text)xxxxxxxxxx3 1headers['Cookie'] =
session2response =
requests.get(url=url2,headers=headers)3print(response.text)
```

7.于是得到flag

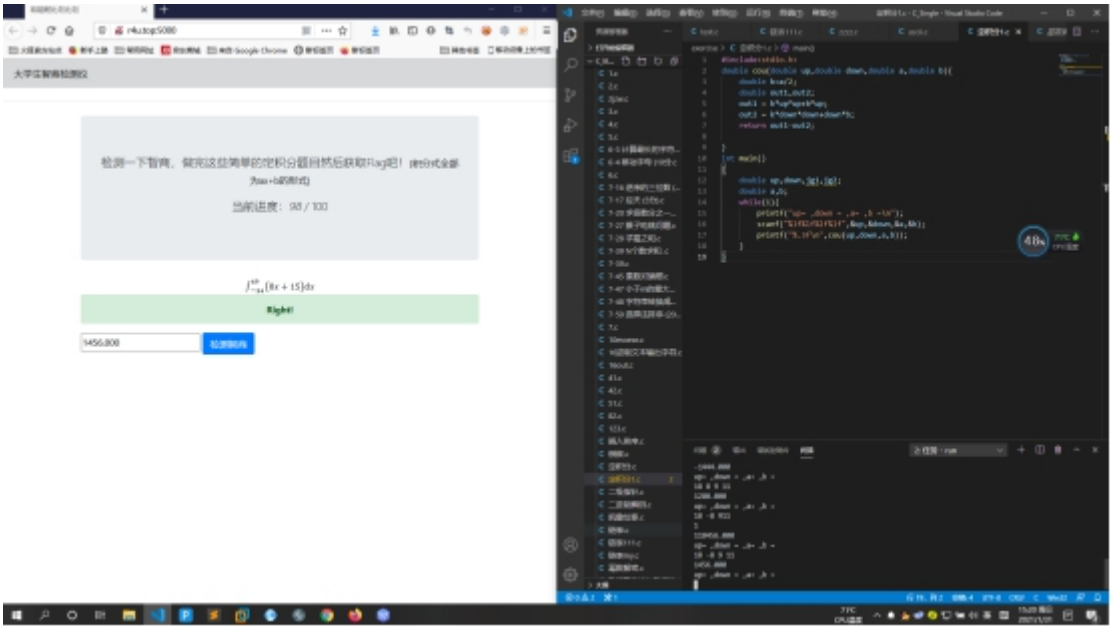


最后总的代码如下

```
import requests,re
import json
url2= 'http://r4u.top:5000/api/getFlag'
url1= 'http://r4u.top:5000/api/getStatus'
url = 'http://r4u.top:5000/'
geturl = 'http://r4u.top:5000/api/getQuestion'
posturl = 'http://r4u.top:5000/api/verify'
session = 'session=eyJzb2x2aW5nIjowfQ.YB4xhQ.JDbHS_sSuPcUhlofoNDyo-j339U'
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0",
    "Accept": "application/json, text/javascript, */*; q=0.01",
    "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-
US;q=0.3,en;q=0.2",
    "Accept-Encoding": "gzip, deflate",
    "Content-Type": "application/json;charset=utf-8",
    "Content-Length": "14",
    "Origin": "http://r4u.top:5000",
    "Connection": "keep-alive",
    "Referer": "http://r4u.top:5000/",
    "Cookie": session,
}
data = {
    'answer':'  '
}
def getQuestion():
    response = requests.get(url=geturl,headers=headers)
    res = response.text
    print(res)
    p = re.compile(r'([\d]{1,5})<')
    p1 = re.compile(r'(-).*?([\d]{1,5})<')
    p_res = p.findall(res)
    p1_res = p1.findall(res)
    #print(p_res)
    #print(p1_res)
    for i in range(0,len(p1_res)):
        for j in range(0,4):
            if(p_res[j])==p1_res[i][1]:
                p_res[j]='-'+p_res[j]
                p1_res[i]=('','9999')
```

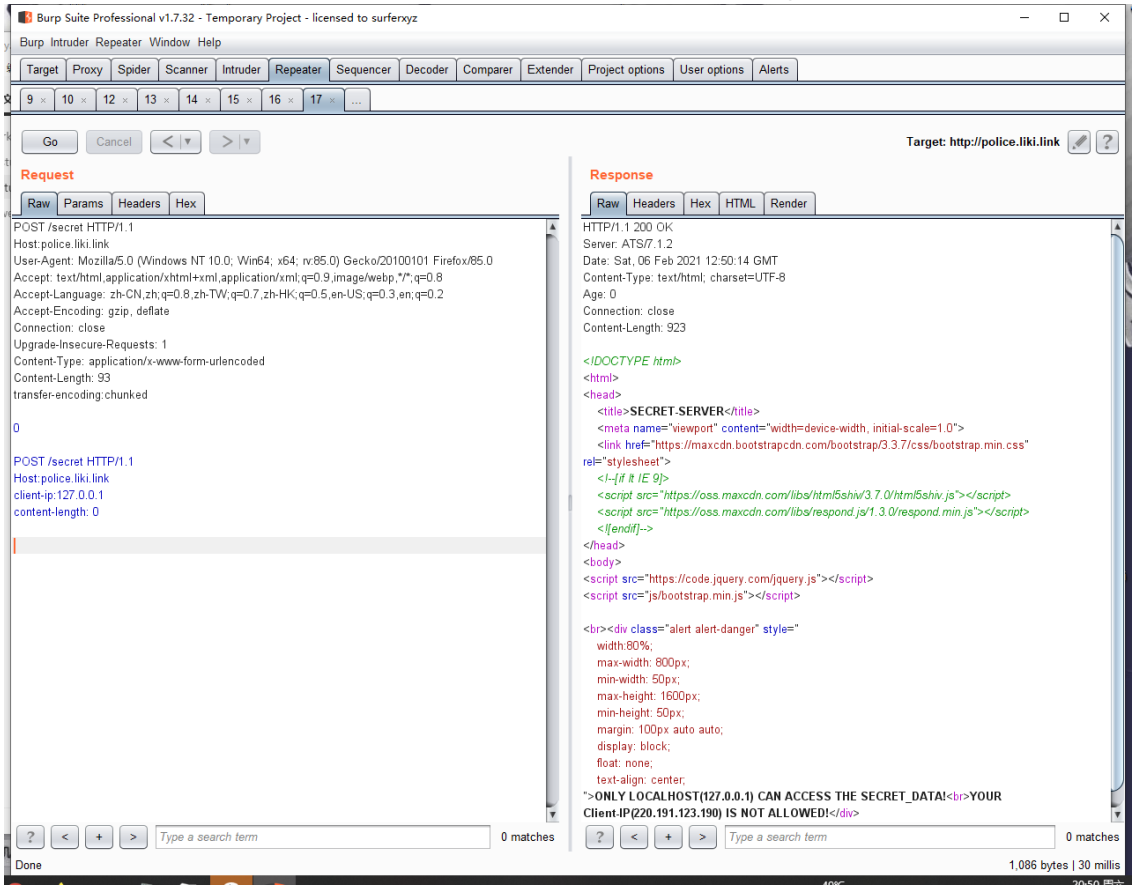
```
    run = [float(x) for x in p_res]
    print(run)
    return run#返回列表对象，列表对象里面是取得的几个问题数字

def counts(t,x,c):#用于计算
    return 0.5*x*t**2+c*t
def solveitone(question):
    down = question[0]#积分上限
    up = question[1]#积分下限
    x = question[2]#一次项
    c= question[3]#常数项
    a = counts(up,x,c)#计算积分上限值
    print(a)
    b = counts(down,x,c)#计算积分下限值
    print(b)
    answer = a-b#减一减得出定积分答案
    print(answer)
    return answer #返回一个数字，是要post的答案

def postAnswer(data):
    response = requests.post(url=posturl,data=data,headers=headers)
    print(response.text)
    head = response.headers
    x = requests.structures.CaseInsensitiveDict(head)
    js = dict(x)
    session = js['Set-Cookie']
    return session#返回一个字符串对象，是Set-Cookie的值，用于判断题目第几题
    #就是把答案post过去，然后可以更新cookie的值，然后就可以得到下一题的题目信息
i=0
for i in range(0,100):
    headers['Cookie'] = session
    question = getQuestion()
    answer = solveitone(question)
    data['answer'] = answer
    session = postAnswer(json.dumps(data))

headers['Cookie'] = session
response = requests.get(url=url2,headers=headers)
print(response.text)
```

写了个c语言程序(我的爬虫每次都post失败555)



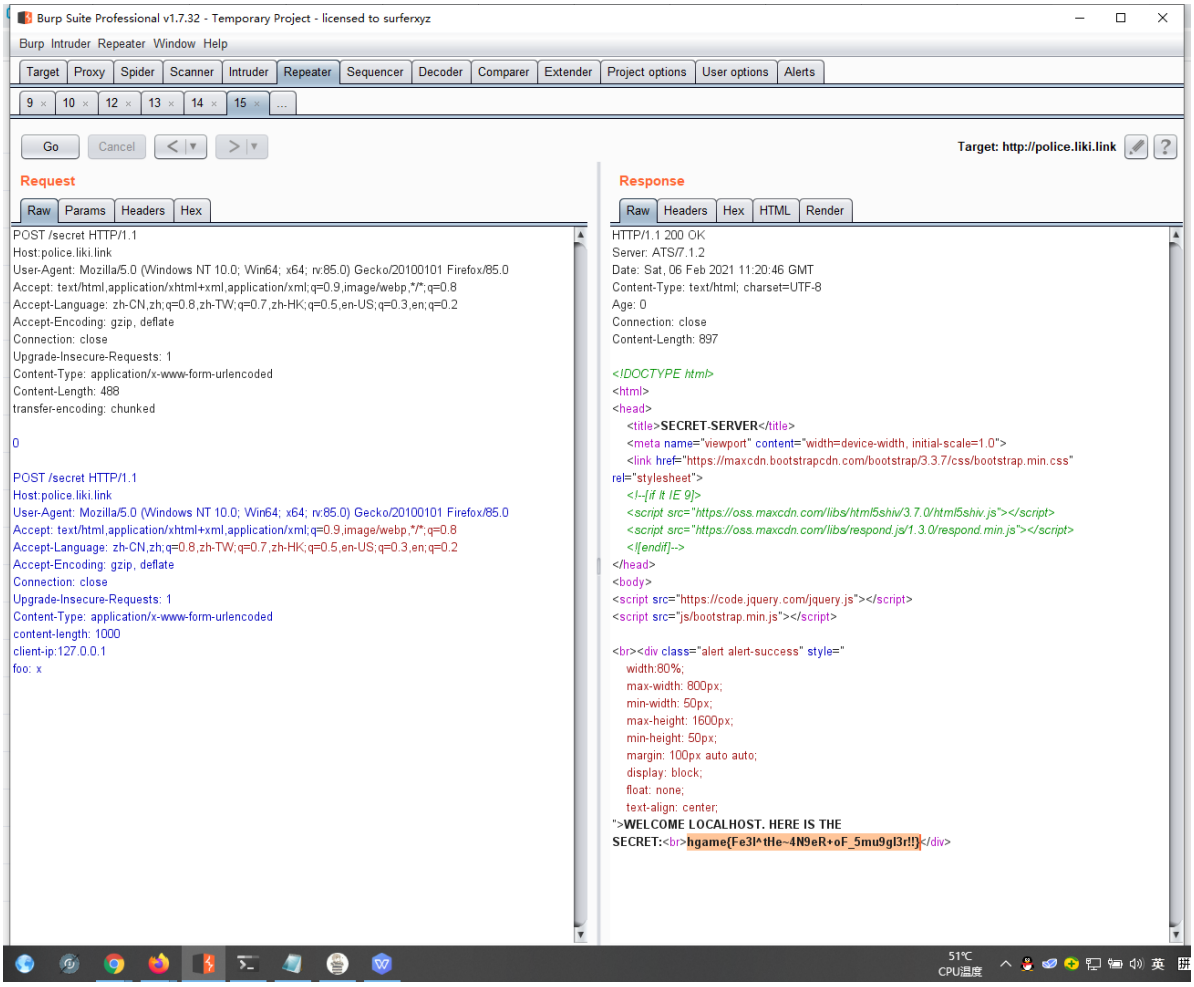
5. 愤怒的走私者

1. 与第三题类似，这是一个cl-te结构，考虑可以像第三题一样，进行走私攻击
2. 发送了和三一样的请求，发现不行，猜测可能在服务器里有对client-ip的过滤



3. 于是考虑在post的请求里再发送一个cl:10000，
4. 因为我发送的数据不到1000，又因为是持久的pipeline连接，所以服务器等待时把后来传输的一些数据加到这一次的数据后面
5. 发送第二次请求，这一次的请求会添加在上一次请求的后面，因为上一次请求事迹发送的cl不到1000
6. 于是这一次发送的数据通过pipeline走私到了后端服务器
7. 又因为这一次请求正文里的te:0
8. 所以post的数据被当作正常请求，且被认为通过了前端的安全验证

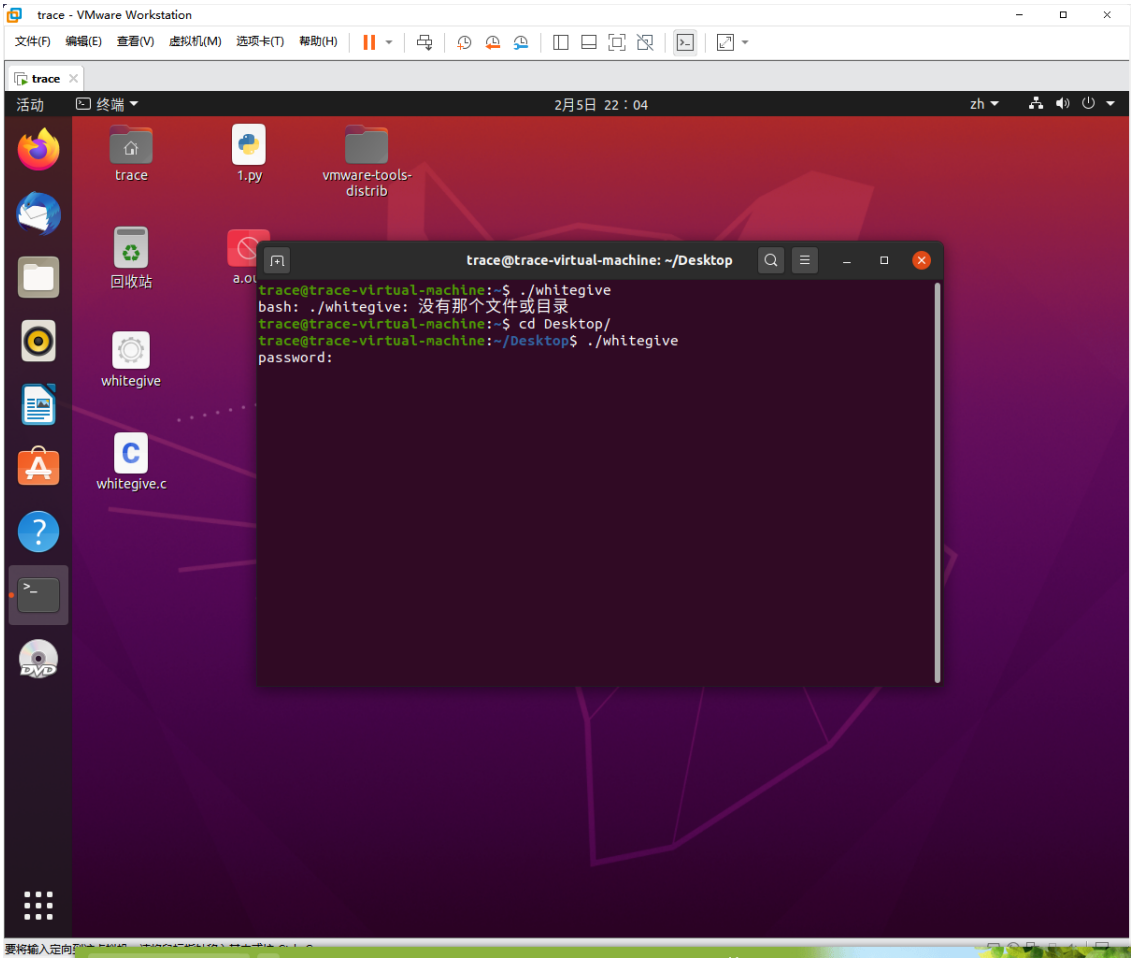
9. 在post里的数据里加入client-ip：127.0.0.1
10. 返回flag



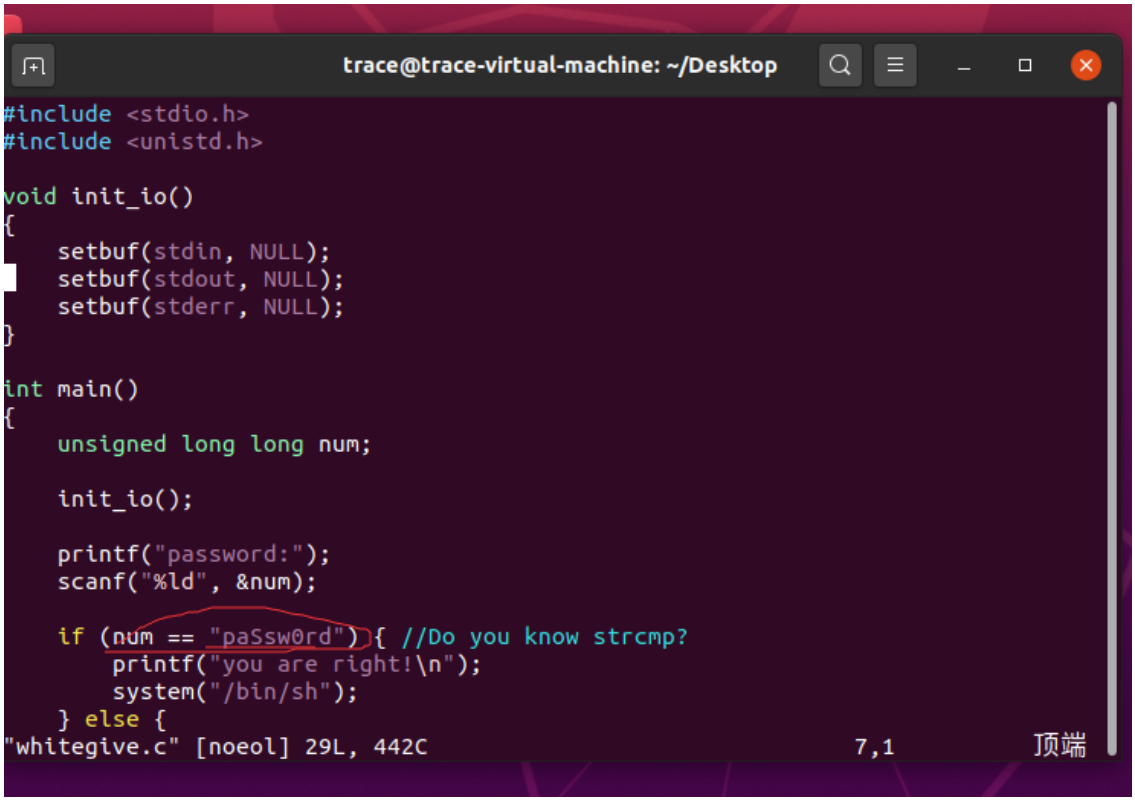
pwn

1. whitegive

1. 打开下载的压缩包，解压后发现是一个elf文件和一个c语言源程序
2. 打开elf文件，发现要我们输password，猜想这里的判定就是靠根据c语言程序来的



3. 阅读c语言代码



4. 得知能不能拿到shell是根据 num和paSsw0rd是否相等来判断的
5. 显示想到strcmp是一个一个字符比较直到最后一个，相等的返回是0，于是先写了个p的ascii上去，发现不对。想到要有\0结尾才行。
6. 于是输了个\0上区，不行。想到可能是ld的原因，输了个0上去，还是不行

7. 百度strcmp，看到strcmp的定义想起这是按内存地址比较的

源码

```
int strcmp(const char *str1,const char *str2)
{
    /*不可用while(*str1++==*str2++)来比较，当不相等时仍会执行一次++，
    return返回的比较值实际上是下一个字符。应将++放到循环体中进行。*/
    while(*str1 == *str2)
    {
                assert((str1 != NULL) && (str2 != NULL));
        if(*str1 == '\0')
            return 0;
        str1++;
        str2++;
    }
    return *str1 - *str2;
}
```

8. 所以如果可以直到elf中这个p开头的内存地址就行了
9. 百度。在csdn找到一个教怎么找内存地址的，其中第二个方法可以用

返回的是地址最低 3 位。因为就算随机化，最低 3 位地址也保持不变。

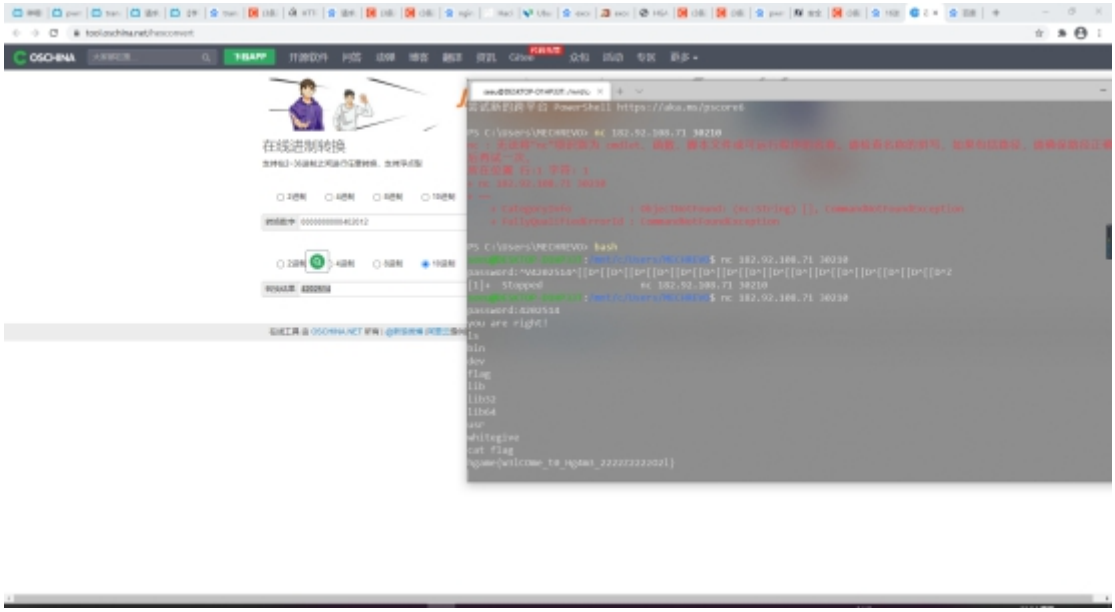
0x2 ROPgadget

```
ROPgadget --binary mypwn --string '/bin/sh'
```

10. 得到地址的16进制

```
0x00000000004020dc : p
0x0000000000403e00 : p
trace@trace-virtual-machine:~/Desktop$ ROPgadget --binary whitegive --string 'paSsw0rd'
Strings information
============================================================
0x0000000000402012 : paSsw0rd
trace@trace-virtual-machine:~/Desktop$
```

11. 转位10进制后输入(因为ld是十进制)，得到shell
12. 直接ls，然后cat flag即可



crypto

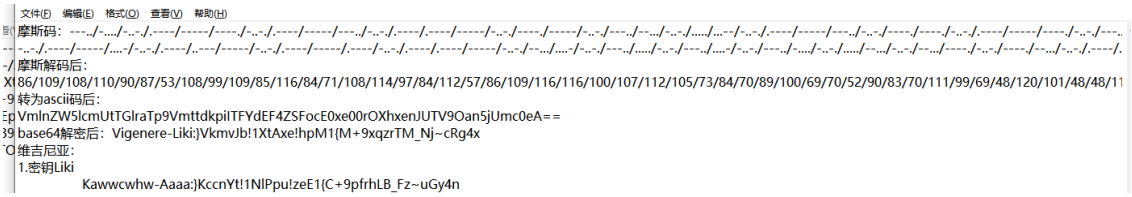
1. cipher

1. 下载得到一个没有后缀的文件，file查看，发现是txt文件

```
seeu@DESKTOP-D1HP33T:/mnt/c/Users/MECHREVO/Downloads$ file cipher
cipher: ASCII text, with very long lines, with no line terminators
seeu@DESKTOP-D1HP33T:/mnt/c/Users/MECHREVO/Downloads$
```

2. 用记事本打开得到摩斯码

   解码后的几个密码都很明显

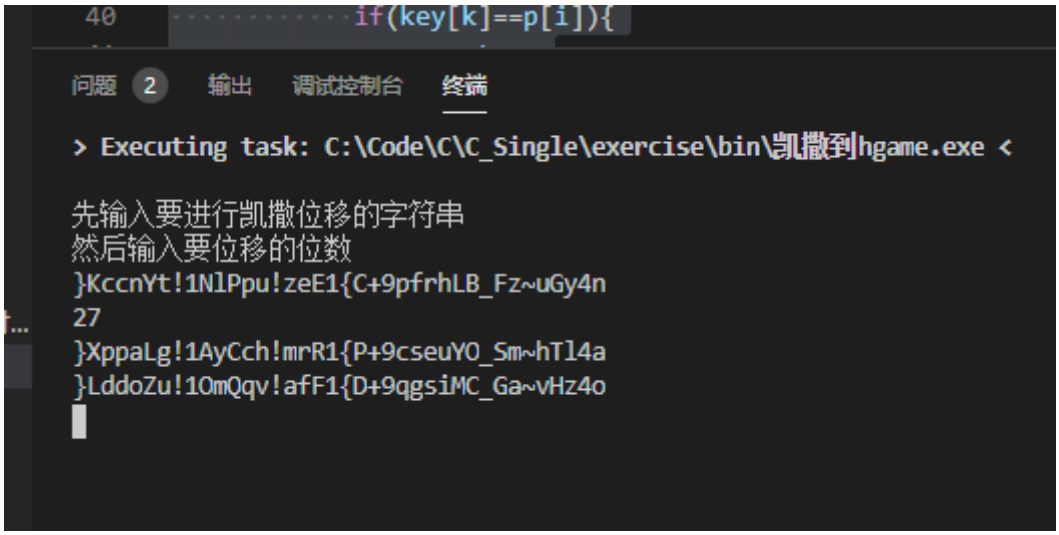


3. 维吉尼亚解码后得到一串奇怪的字符，根据提示，解码后的明文里有hgame字样，于是想到了凯撒位移

4. 但是位移不出来，想到可能hgame几个字符的顺序不一定，于是写了个小脚本跑一下，看看hgame几个字符同时出现的情况

```c
#include<stdio.h>
#include<string.h>
void yd(char *p,int l);
int jduge(char *p);
char key[5]="hgame";
const num = 5;
int main()
{
    char a[103];
    int n,i;
    printf("先输入要进行凯撒位移的字符串\n然后输入要位移的位数\n");
    gets(a);
    while (scanf("%d",&n)!=EOF)
    {
        for(i=0;i<n;i++){
        yd(a,strlen(a));
        }
        for(i=0;i<strlen(a);i++){
        printf("%c",a[i]);
        }
        printf("\n");
    }
}
void yd(char *p,int l)
{
    for(int i=0;i<l;i++){
        if(p[i]!='z'&&p[i]<'z'&&p[i]>='a'||p[i]!='Z'&&p[i]
<'Z'&&p[i]>='A') p[i]=p[i]+1;
        else if(p[i]=='z') p[i]='a';
        else if(p[i]=='Z') p[i]='A';
    }
    if(jduge(p)){
        printf("%s\n",p);
    }
}
int jduge(char *p)
{
```

```
    int i=0,count=0;
    for(int k=0;k<=num;k++){
        for(i=0;i<=strlen(p)-1;i++){
            if(key[k]==p[i]){
                count++;
                break;
            }
        }
    }
    return (count==num);
}
```



5. 得到两个字符串，猜想在这两个中的一个是要的
6. hgame几个字符的顺序不在一起，所以猜测可能是栅栏密码
7. 多次尝试后，发现上面那个字符串的6位栅栏有一个倒着的hgame，想到格式是hgame{xxx}
8. 于是倒过来输进去flag正确