Hgame writeup

WEB

第一题

这道第一题折腾了好久，第一步比较套路，发现标题是where is my robots，需要查看爬虫协议，输入查看，给了一个地址。

```
img/index.php
```

打开发现一张草&&区的图和一堆代码，让我想起了18年应该也有同样一道题有这个图(记不太清orz)。看代码，发现用str_replace过滤了一次../这个和以前做过的一道题类似，只要构造出一个字符串在过滤之后拼接成../就可以实现目录跳转。所以构造..././会跳回到上一层目录，然后我就在这里卡了。输入进去index也不行，看起来没有反应，可能是没有index.php这个文件。然后想着img目录下的index文件，..././img/index取了一次还是没有反应。开始困惑。然后在**下，获得了重要hint，flag.php

(这猜的成分也有点高了吧。)不过我也曾经想过burp suite爆破一下目录2333..././flag果然出来了显示，不过提示maybe_you_should_think_think这儿我也想了一会儿，不直接给flag就意思是在文件里，但是我应该用另一种方法来获取到文件，不能靠它本来的输出，联系之前做出来的PHP Is The Best Language 就可以想到伪协议这个办法(土爷天下第一，博客啥都有)，正好网上也说include_once也能用，所以试一下。但是这里也有个坑。也算是我之前不知道，

正确的payload应该是：

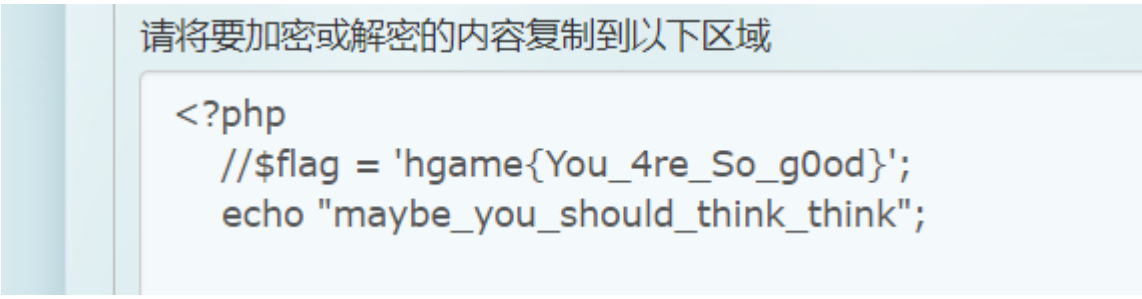
```
php?img=php://filter/read=convert.base64-encode/resource=..././flag
```

但是当时我以为应该是：

```
?img=..././php://filter/read=convert.base64-encode/resource=flag
```

伪协议中应该包含了这个路径的跳转。所以我认为的不正确，也是在偶然之间把目录放在后面，发现可以的，这一点应该记下来。

最后就得到了base64的文件内容，解密以后发现flag是作为注释存在的。 `Payload: http://118.24.25.25:9999/easyphp/img/index.php?img=php://filter/read=convert.base64-encode/resource=..././flag`



第二题

php trick

在出来结果的一瞬间 胸中万千只草泥马在奔腾 总结一下知识： 最开始的

step 1，step2。

Md5开头为0的绕过

step3，step4

变量为数组的直接为0（弱类型比较）

step5，step6，step7，step8

url编码绕过字符串存在的检查(strpos) `%48%5f%67%61%6d%65` 就是H_game

不过看同学的解决方式中存在，`H+game` 来达到效果的方法，原因如下。

> 在遇到需要在url中请求如：A_A这样的值的时候，可以通过A.A或者A+A来达到相同的效果。
>
> **PHP变量名不能带有点[.]和空格，否则在会被转化为下划线[_]**

第七，第八不清楚咋绕过去的 但是把hgame设置成数组之后就绕过去了，网上查了个资料。不清楚详细的操作。

第十一题：弱类型整数大小比较绕过

```
1 $temp = $_GET['password']; is_numeric($temp)?die("no numeric"):NULL; if($temp>1336){ echo $flag;
```

分析：传入password赋值给temp，is_numeric判断temp是不是数字，是则die，不是数字就继续向下运行，再就是传入的值必须大于1336，上边提到过，数组在比较中恒大于具体的值，并且数组不是数字或者文本型数字，可以绕过数字检测。
参考：?password[]=

第九第十特别坑，把要求输进去，然后发现给打开了百度，但是明显要的不是百度，而是本地的上面已经提示的admin.php文件，接下来就是一直憋一直查，然后发现了一个curl与parse_url对于地址分割的不同。(不过听群上老大哥的意思是这个漏洞在后来的版本中修复了？)

0×01 SSRF

通过代码逻辑我们可知

> url->php parse_url（过滤ip）->过滤url各部分(空白字符和数字)->curl发送请求

这里可利用`parse_url`和`libcurl`对url解析的差异来绕过。经过测试,得出以下结论（我本地环境`php 7.0.20-2 libcurl/7.52.1`）

```
完整url: scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]
这里仅讨论url中不含'?'的情况

php parse_url:
host: 匹配最后一个@后面符合格式的host

libcurl:
host: 匹配第一个@后面符合格式的host

如:

http://u:p@a.com:80@b.com/

php解析结果:
    schema: http
    host: b.com
    user: u
    pass: p@a.com:80
libcurl解析结果:
    schema: http
    host: a.com
    user: u
    pass: p
    port: 80
    后面的@b.com/会被忽略掉
```

parse_url与libcurl对与url的解析差异可能导致ssrf

- 当url中有多个@符号时，**parse_url中获取的host是最后一个@符号后面的host，而libcurl则是获取的第一个@符号之后的。**因此当代码对`http://user@eval.com:80@baidu.com`进行解析时，PHP获取的host是baidu.com是允许访问的域名，而最后调用libcurl进行请求时则是请求的eval.com域名，可以造成ssrf绕过

- 此外对于`https://evil@baidu.com`这样的域名进行解析时,php获取的host是`evil@baidu.com`，但是libcurl获取的host却是evil.com

url标准的灵活性导致绕过filter_var与parse_url进行ssrf

filter_var()函数对于`http://evil.com;google.com`会返回false也就是认为url格式错误，但是对于`0://evil.com:80;google.com:80/`、`0://evil.com:80,google.com:80/`、`0://evil.com:80\google.com:80/`却返回true。

解决以后就可以以本地用户(127.0.0.1)的身份访问之前无法访问的admin.php 这里就是一个目录，访问以后就出现了进一步的代码。提示还有一个flag.php 但是继续观察发现给了一个变量，这里的??运算符我还不清楚还查了一下，这里就是检测是否变量赋值 如果没有赋值的话就赋值为空，当然我们想要获取flag文件啦，但是这里有一个if，如果文件存在，就报错，不存在才给输出。。这不是自相矛盾。。。。纠结了好长时间，查了一下，然后想了想咋就能输出文件内容，以前也接触过伪协议，这次尝试着用base64输出文件内容，没想到就过了检测，直接输出了字符串，然后就直接拿着字符串去解码就可以了


```
名称    描述
resource=<要过滤的数据流>    这个参数是必须的。它指定了你要筛选过滤的数据流。
read=<读链的筛选列表>        该参数可选。可以设定一个或多个过滤器名称，以管道符（|）分隔。
write=<写链的筛选列表>       该参数可选。可以设定一个或多个过滤器名称，以管道符（|）分隔。
<; 两个链的筛选列表>         任何没有以 read= 或 write= 作前缀 的筛选器列表会视情况应用于
```

我们举一个例子，这是平时我们用来任意文件读取的payload

```
php://filter/read=convert.base64-encode/resource=upload.php
```

这里读的过滤器为convert.base64-encode，就和字面上的意思一样，把输入流base64-encode。resource=upload.php，代表读取upload.php的内容

下面仔细研究下关于过滤器的问题

过滤器

先贴译档，不因为自己的翻译小问题接锅 (・ω・)ノ
http://php.net/manual/zh/filters.php

转换过滤器

http://php.net/manual/zh/filters.convert.php

convert.* 过滤器是php5.0.0以后添加的。

base64

convert.base64-encode和 convert.base64-decode使用这两个过滤器等同于分别用 base64_encode()和

http://118.24.3.214:3001/index.php?str1=s155964671a&str2=s214587387a&str3[]=1&str4[]=2&%48%5f%67%61%6d%65[]=%3&url=http://@127.0.0.1:80@www.baidu.com/../admin.php?filename=php://filter/read=convert.base64-encode/resource=flag.php

第三题

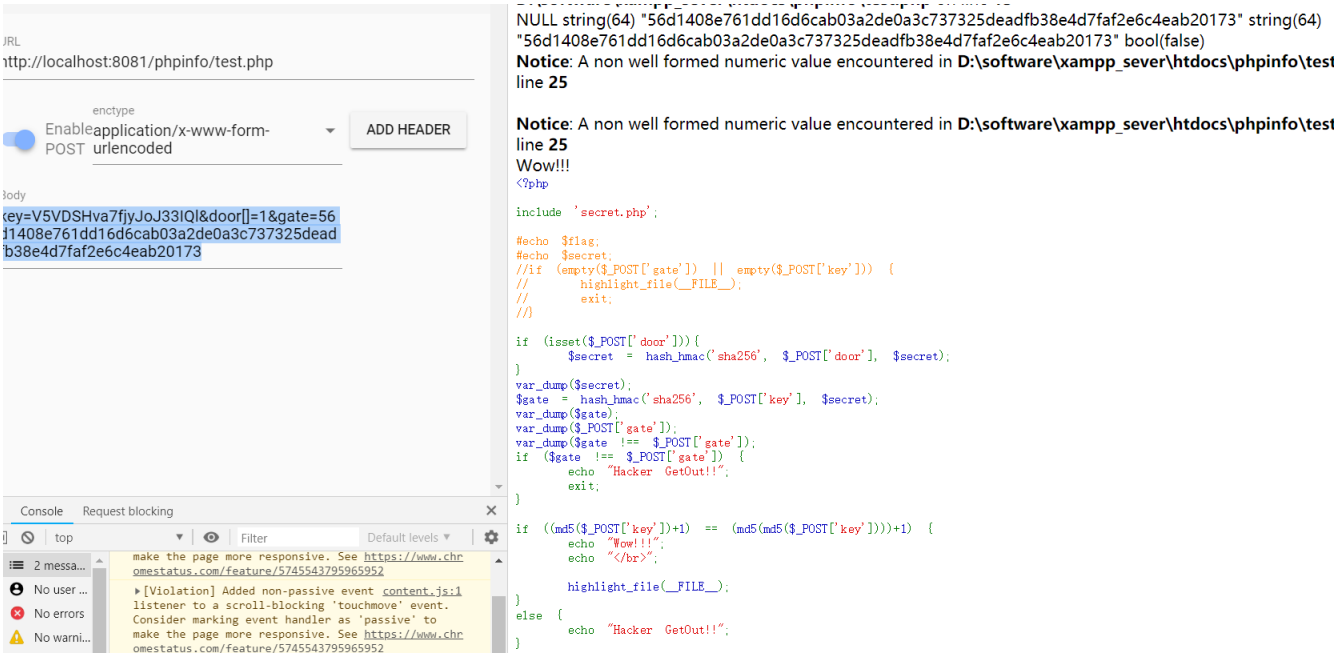
这道题是`BrownFly`学长手把手教的，感谢学长的不嫌弃。

题目描述：var_dump了解一下

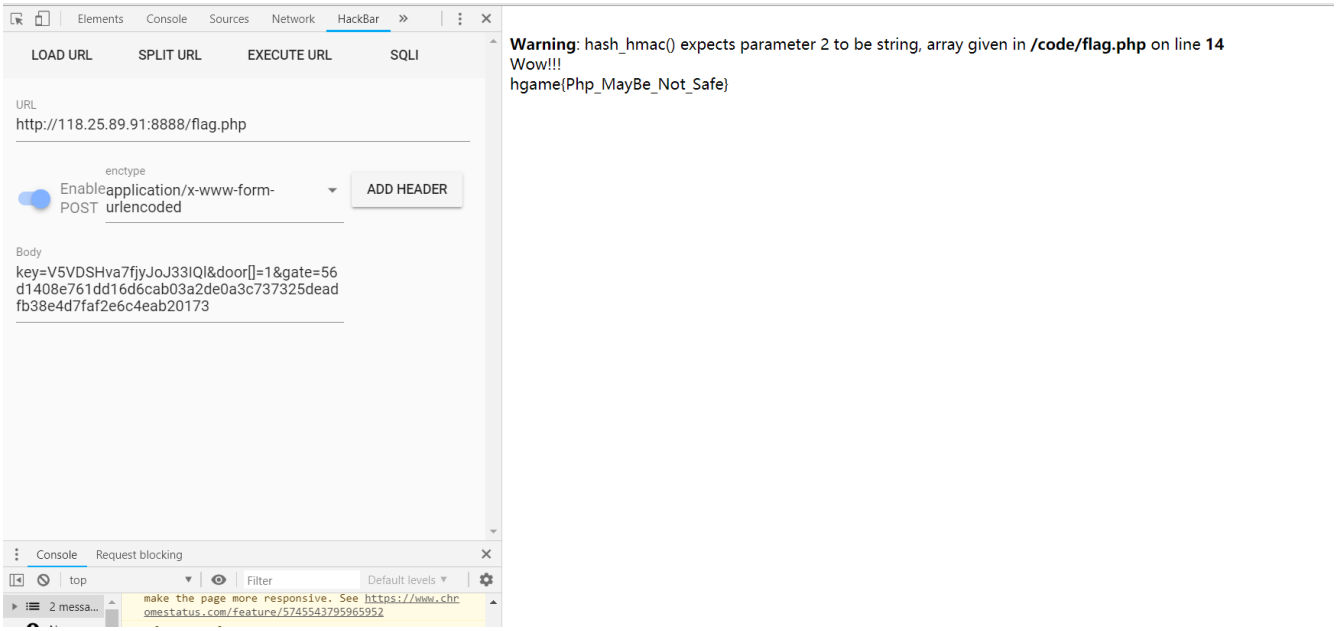
真正问了学长以后才重视这个函数，期间自己搭了一个服务器用来测试php，用var_dump能够输出当前变量的数据类型与当前的值，对于===这种强类型比较来说很有用，就可以判断数据两边的数据类型。

整理一下题的思路:

分析第一次的比较过程，其中存在一个随机的加密密匙，`$secret`对于这个因素我们是不可控的，所以要在第二步的重设密匙中寻求方法来重置密匙。这里存在一个方法，把door设成数组，这样在hash_hmac()函数中第二个参数就成了数组，应该是默认会处理为空？用var_dump输出一下，发现返回值为NULL所以说第二次的加密过程中的密匙就可以预测了，不过也不需要知道具体的密匙是多少，直接在php文件中echo出来结果即可。这样处理之后，就可以根据key的明文值，知道gate对应的密文值了。



绕过两次比较的关键点是，key的取值，我一开始的想法是找到一个两次md5加密均为0e开头的key值，然后想想有些疑惑，这里有一个弱类型比较，看起来也可以通过key取数组来达成，但是这里如果key取数组就会产生一个问题，第三步gate的值就会为NULL，而我们输入的gate值在第一步判断中不能为空（我也不清楚如果放开了第一步的判断能不能输入一个空值，忘记试了。）所以这条路就断了，只能寻求下一个可能（学长手把手提醒），这种两次加密均为0e开头的字符串还真存在，网页上扣下来，并且扔进php中计算出gate应取的值，填进去即可。



`Payload:`

`key=V5VDSHva7fjyJoJ33IQl&door[]=1&gate=56d1408e761dd16d6cab03a2de0a3c737325deadfb38e4d7faf2e6c4eab20173`

`V5VDSHva7fjyJoJ33IQl` 双md5结果均为 `0e`

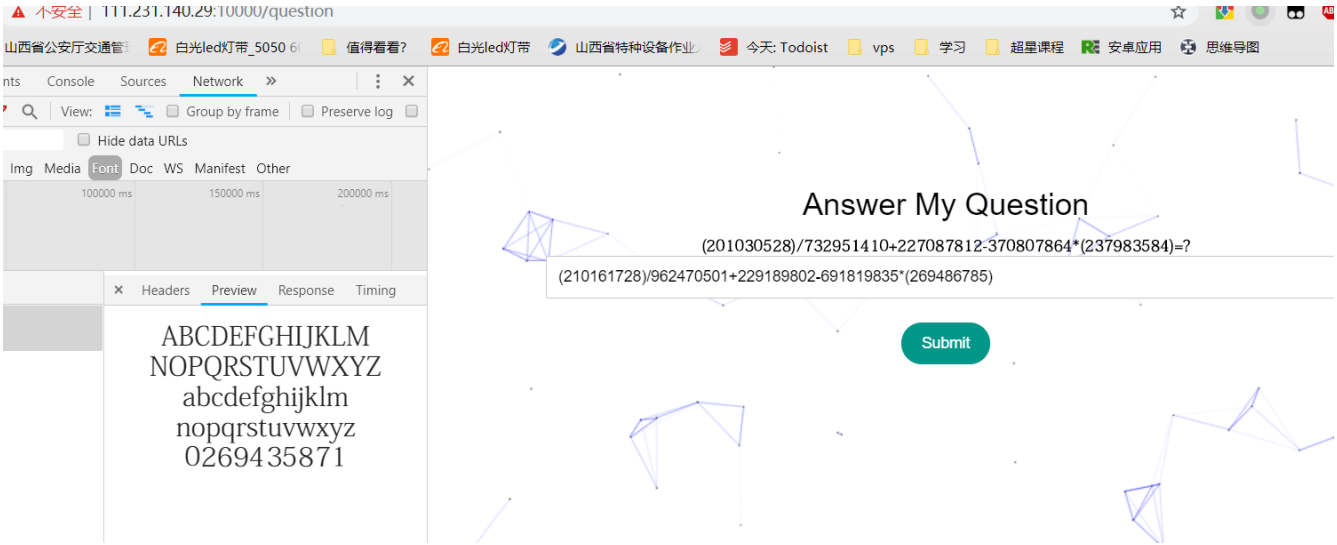
第四题

Baby_Spider

li4no学长nb！！！

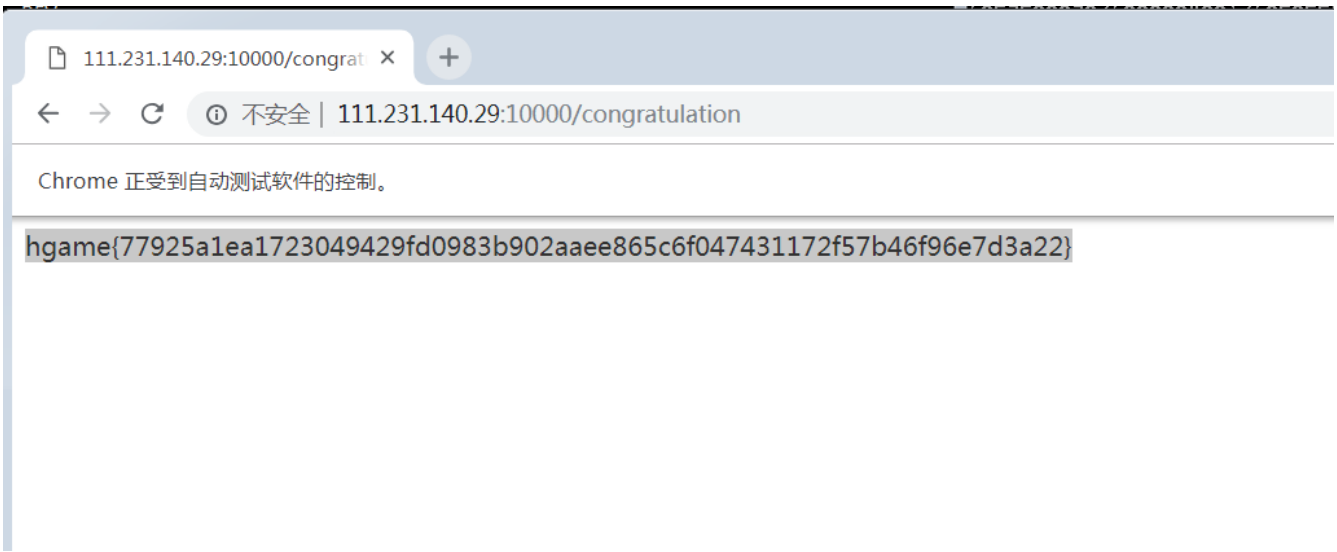
共有3个关口，第一次是缺user-agent，会让你关机，亏得我机没关完，还有几个程序垂死挣扎，所以就没完全关掉。亏得我保存了。所以海星。。再也不敢了。加入user-agent即可避免，但是接下来是下一个关口，还是错误答案。

第二个关口是加入了一个字体，需要更改对应的数字来完成计算



这个是我手动做完10道题获取到的，当时复制下来的公式与网页上显示的公式不同，这就让我很困惑，最后在浏览器的网络界面，发现加载了一个新的字体文件，所以打开这个文件，按照数字一一对应关系来更改从代码中直接获取的数字，就可以继续完成计算。

第三个关口到达条件就比较艰难，因为一开始写的爬虫是基于原始的session库的，并不能达到获取全部文件的功能(即使有我也没学过)，手动根本做不到第20题，也就没办法得到进一步的网页界面更改的信息，就改变了做法，通过selenium这个库来进行操作，它可以直接控制浏览器的操作，不仅仅能通过爬虫的基础检测，还能实时查看加载的各种文件，直接在网页上发现更改。21-30题用了css的伪类，这个知识点我也不太清楚，但是需要根据css文件来获得式子的具体值。我用js来获取相关的伪类中的数据，之后返回给python，利用python来计算值之后按照之前的方法输出，最后得到flag。



payload:

```
import time
from selenium import webdriver
chrome = browser = webdriver.Chrome()
url = 'http://111.231.140.29:10000/'
post = {'token':'e5F8rkjyqj3ohhpVtP2HGPgR8bf8ZjMV'}
chrome.get(url)
token = chrome.find_element_by_name('token')
token.send_keys('e5F8rkjyqj3ohhpVtP2HGPgR8bf8ZjMV')
submit = chrome.find_element_by_css_selector('button')
submit.click()

for i in range(10):
    expression = chrome.find_element_by_css_selector('span').text
    print(expression[:-2])
    expression = expression[:-2]
    answer = chrome.find_element_by_name('answer')

    answer.send_keys(str(eval(expression)))
    submit = chrome.find_element_by_css_selector('button')
    submit.click()

for i in range(10):
    expression = chrome.find_element_by_css_selector('span').text
    print(expression[:-2])
    expression = expression[:-2]
    for i in range(len(expression)):
            if (expression[i] == '1'):
                b = expression[:i]+'0'+expression[i+1:]
                expression = b
                continue
            if (expression[i] == '0'):
                b = expression[:i]+'1'+expression[i+1:]
                expression = b
                continue
            if (expression[i] == '3'):
```

```
                b = expression[:i]+'6'+expression[i+1:]
                expression = b
                continue
            if (expression[i] == '4'):
                b = expression[:i]+'9'+expression[i+1:]
                expression = b
                continue
            if (expression[i] == '5'):
                b = expression[:i]+'4'+expression[i+1:]
                expression = b
                continue
            if (expression[i] == '6'):
                b = expression[:i]+'3'+expression[i+1:]
                expression = b
                continue
            if (expression[i] == '7'):
                b = expression[:i]+'5'+expression[i+1:]
                expression = b
                continue

            if (expression[i] == '9'):
                b = expression[:i]+'7'+expression[i+1:]
                expression = b
                continue
    answer = chrome.find_element_by_name('answer')

    answer.send_keys(str(eval(expression)))
    submit = chrome.find_element_by_css_selector('button')
    submit.click()

#expression = chrome.execute_script("windows.getComputedStyle(span,'::after').content")
#expression = chrome.execute_script('return
window.getComputedStyle(document.querySelector(".question-
container"),"::after").content;')

for i in range(10):
    expression = chrome.execute_script('return
window.getComputedStyle(document.querySelector(".question-
container"),"::after").content;')
    expression = expression[1:-3]
    answer = chrome.find_element_by_name('answer')
    answer.send_keys(str(eval(expression)))
    submit = chrome.find_element_by_css_selector('button')
    submit.click()
```

re

第五题

re只会第五题 555~

Pro的Python教室

网页有相关的pyc在线逆向工具，扔进去得到逆向的结果。

```
#!/usr/bin/env python
# encoding: utf-8

enc = 'hgame{xxxxxxxxxxxxxxxxxxxx}'
len = len(enc)
enc1 = []
enc2 = ''
aaa = 'ioOavquaDb}x2ha4[~ifqZaujQ#'
for i in range(len):
    if i % 2 == 0:
        enc1.append(chr(ord(enc[i]) + 1)) #偶数+1
        continue                          #奇数+2 对ascii
    enc1.append(chr(ord(enc[i]) + 2))

s1 = []
for x in range(3):      #X= 0, 1, 2
    for i in range(len):
        if (i + x) % 3 == 0:
            s1.append(enc1[i])
            continue
    # x = 0时
    # 位数为3倍数的字符 0 3 6 9 12 15 18 21 24
    # 加入s1
    # x = 1时
    # 位数为1 4 7 10的字符
    # 加入s1
    # x = 2时
    # 位数为2 5 8 11的字符
    # 加入s1



enc2 = enc2.join(s1)


if enc2 in aaa:
    print "You 're Right!"
else:
    print "You're Wrong!"
    exit(0)
```

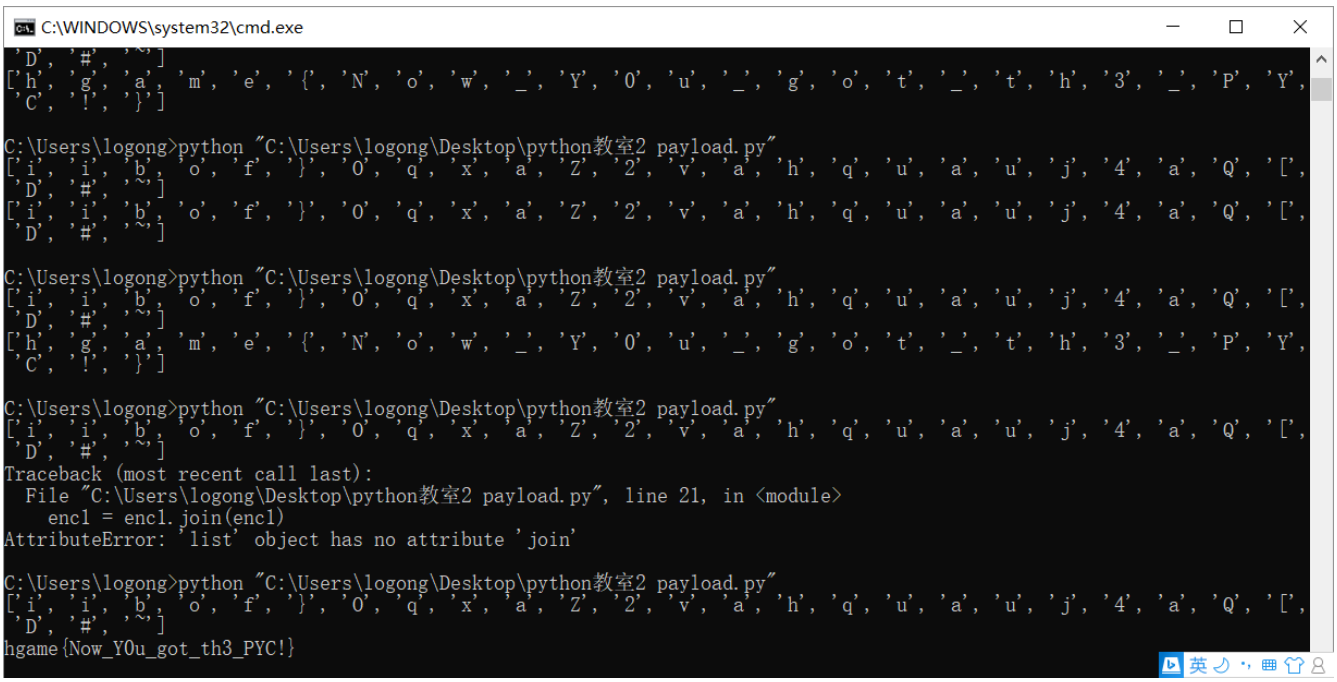
有代码的话就好解决了，对应它的操作来进行真正的·逆向·

```
aaa = 'ioOavquaDb}x2ha4[~ifqZaujQ#'
#print(len(aaa))  #字符串长度为27 x in range3 所以一次处理9个字符 推测1
x1 = aaa[:9]
```

```
x2 = aaa[9:18]
x3 = aaa[18:]
b = [0]*27
for i in range(9):
    b[i*3]=x1[i]
for i in range(9):
    b[i*3+1]=x3[i]
for i in range(9):
    b[i*3+2]=x2[i]
print(b)

enc1 = []
enc2=''
for i in range(27):
    if i%2 == 0:
        enc1.append(chr(ord(b[i])-1))
        continue
    enc1.append(chr(ord(b[i])-2))
enc2 = enc2.join(enc1)
print(enc2)
```

运行就能得到对应的结果。（发现自己python水平是真的低，大佬们轻喷orz）



pwn

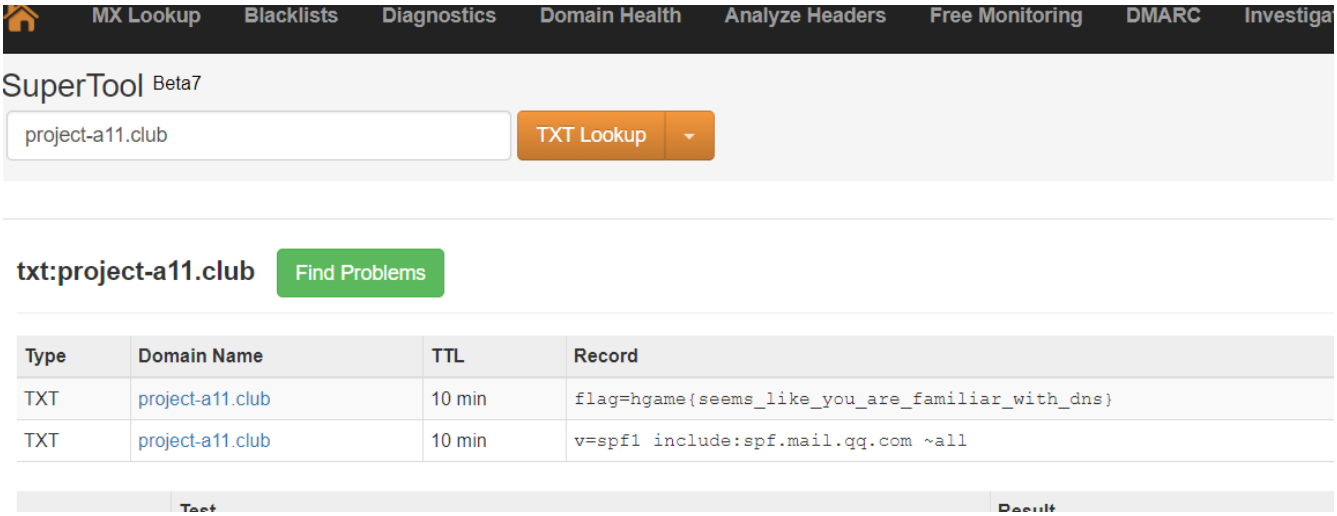
啥都没做出来。

misc

第一题

发现了google的用处，百度出来的dns查询网站没有一个给力的。

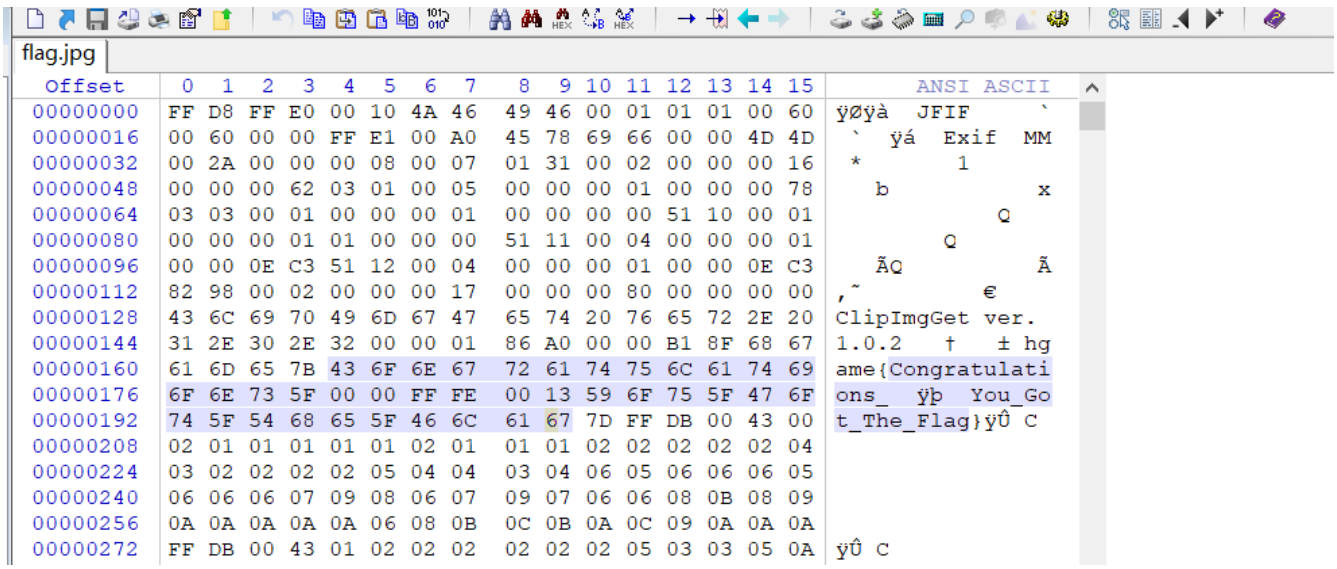
点开这个链接发现啥都没有，而且提示找不到ip，通过放出来的hint了解了dns有很多的种类，



去google了一个靠谱的网站，一个一个一个选项的尝试，最后终于找到了。。

第三题

看了提示是https，给的是pcapng，wireshark的包，打开看一下，没有啥特别的东西，下载能够解析出来的http报文里包含的文件，发现只有一个index网页，而且网页上明确指出flag不在这里。结合hint，查了资料了解了这是一个https的加密方式，需要找到一个log文件用作wireshark解密，就可以分析出https报文内容，那么问题就在于这个文件。报文里还有ftp的一些操作，从中提取报文数据可以还原出一个secret.zip的文件，解压发现正好有一个log文件，于是导入wireshark的设置，就可以解密ssl加密的https报文，发现是一个.tar文件，解压发现一张美美的小姐姐，扔进winhax发现flag。

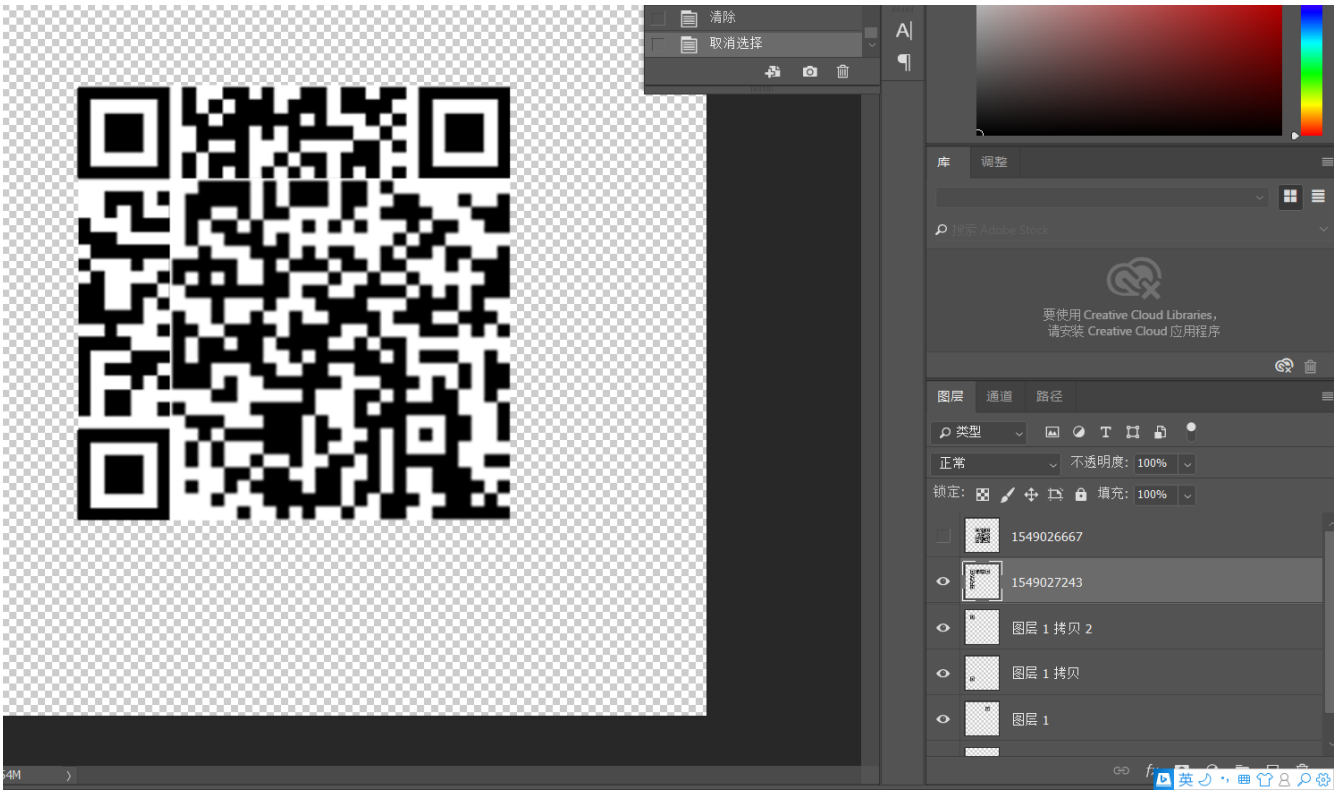


第四题

这道题挺简单，不过费工夫，给的压缩包里有flag.txt，看一下就知道是html中的图片信息，拖进自己的网页，打开发现是一张二维码。挪下来扔进ps。

根据自己已知的二维码知识，这是因为没了关键的识别点而导致不能识别，缺了三个定位点。那就加上，但是问题在于这个二维码的大小，二维码有不同的规格，随着规格的不同，定位点的位置也就不同。就开始找。。。我用的方法就是比较，一个一个数右下角定位点与上边界与左边界的距离，然后找对应的二维码定位点贴上，为此还去网站上做了一个二维码。

拿来对比了一下 然后把这张图上的识别点切过来。扫一下就有了。（多试几次）

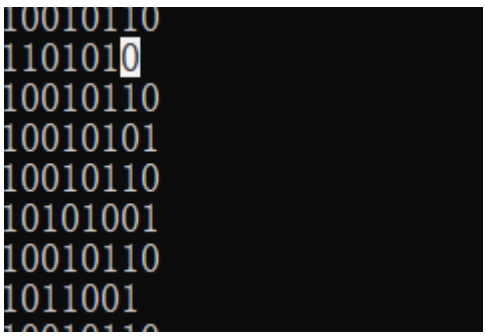


crypto

第一题

浪漫的足球圣地

做了很长时间，看到题目直接查，发现是曼彻斯特，就看一下有没有相关的解码网站，发现一个解码的网站，但是只能解析二进制的数据，于是就写了一个16进制转二进制的脚本，但是脚本写的有问题，曼彻斯特编码一定要4位4位的来，但是转进制以后就出了问题，居然出现了7位，后来研究了一下一个解码工具，发现那个7位的是python自动省略了前缀0，导致缺了位数，缺了位数就不能正确的解析，最后直接在解码工具上把16进制解码了出来，网上找了一个网站转换成字符串就得到flag



第三题

普通的Vigener

这道题，本来很难，纠结了很长时间看不懂维吉尼亚密码要如何来设计脚本来解决。直到发现了一个网站:

扔进去即可，得到flag

##