


web1——谁吃了我的flag

其实题目描述没有好好关机就在暗示了，后来补充hint说vim编写页面就更显然了。vim编写页面非正常关闭时会留下.swp文件。

其实一开始想的就是这个方向但是太菜了只知道往url后面加.swp于是没反应，然后走偏去抓包看见etag这个响应头字段就各种找资料研究http协议的缓存策略，虽然时间拖得很久导致这道题很晚做出来，但也学到了蛮多别的东西。

回归本题正确的做法：除了在url尾部加上.swp还要在文件前加上.



118.25.111.31:10086/.index.html.swp

这样下载下来swp的临时文件打开就可以帮Mki学长找到他被吃掉的flag

web2——换头大作战

想要flag嘛：

request method is error.I think POST is better

f12没有找到有用的提示信息，于是就回来在单行文本框里随便填些内容然后提交。看到随后打印的文字看来要改请求方式为POST。发送到burpsuite的repeater模块里change request method，然后之后按要求一步步改请求报文的首部字段最后就到手flag了。

web3——very easy web

```
<?php
error_reporting(0);
include("flag.php");

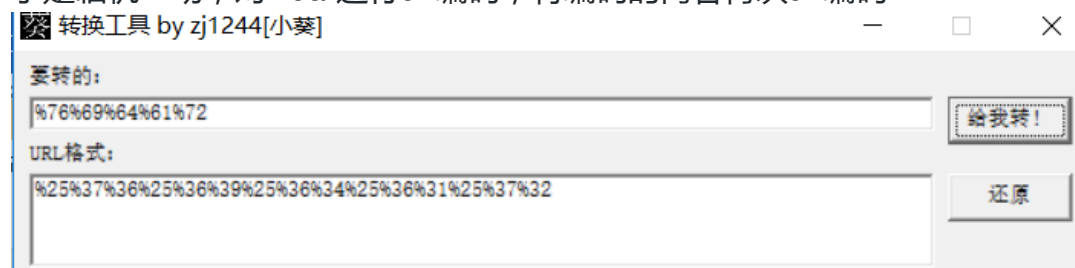
if(strpos("vidar",$_GET['id'])!==FALSE)
    die("<p>干巴爹</p>");

$_GET['id'] = urldecode($_GET['id']);
if($_GET['id'] === "vidar")
{
    echo $flag;
}
highlight_file(__FILE__);
?>
```

确实不难，代码逻辑很简单。利用字符串截取函数判断传入的id值不能包含vidar这个字符串，但是id值url解码之后内容又必须是vidar。

但是对vidar做一次url编码然后跟在?id=后面提交肯定是不行的（因为这就是相当于给id赋值vidar，而我们希望的却是id赋值vidar的url编码）

于是临机一动，对vidar进行url编码，将编码的内容再次url编码



web4——can you find me

点开f12发现href属性值指向f12.php

点击跳转过去，要求post一个password变量。burp suite启动（并不是，刚刚打开就没关

发现是请求方式是GET，依旧发送到repeater模块里change request method。

发现响应报文有个password：woyaoflag

于是在请求报文内容处填password=wayaoflag

发现有个iamflag.php的链接

但是直接点击iamflag.php新页面却提示说我跑太快(莫非学长在暗示我要从网安转到中文，做香港记者比搞安全不知道高哪里去)，把flag丢在某个地方，然后仔细看发现此时的url是toofast.php

所以我猜测从f12.php到iamflag.php之后一定是iamflag.php被重定向到toofast.php

抓包发现果然如此，iamflag的返回报文给了flag，而且头部字段里有location: toofast.php

```
HTTP/1.1 302 Found
Server: nginx/1.15.8
Date: Thu, 31 Jan 2019 06:50:45 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.2.14
location: toofast.php
Content-Length: 132
```

```
<html>
  <head>
    <title>can you find me?</title>
  </head>
  <body>
    <p>flag:hgame{f12_1s_aMazIng111}</p>
  </body>
</html>
```

re2——HelloRe

签到题，丢某个图标为迷人女子的软件里再搭配和做web题必修的F12大法相对应的F5大法就可以直接找到flag

```
v8 = __readfsqword(0x28u);
*(_QWORD *)s = 0LL;
v5 = 0LL;
v6 = 0LL;
v7 = 0LL;
puts("Please input your key:");
fgets(s, 32, stdin);
s[strlen(s) - 1] = 0;
if ( !strcmp(s, "hgame{Welc0m3_t0_R3_World!}") )
    puts("success");
else
    puts("failed..");
return 0LL;
```

re4——pro的python教室

也是签到题，给了三个字符串。代码逻辑反推一下就是，第二个字符串要经过base64解码，第三个字符串要base32加密处理后的三个字符串组合起来就是flag

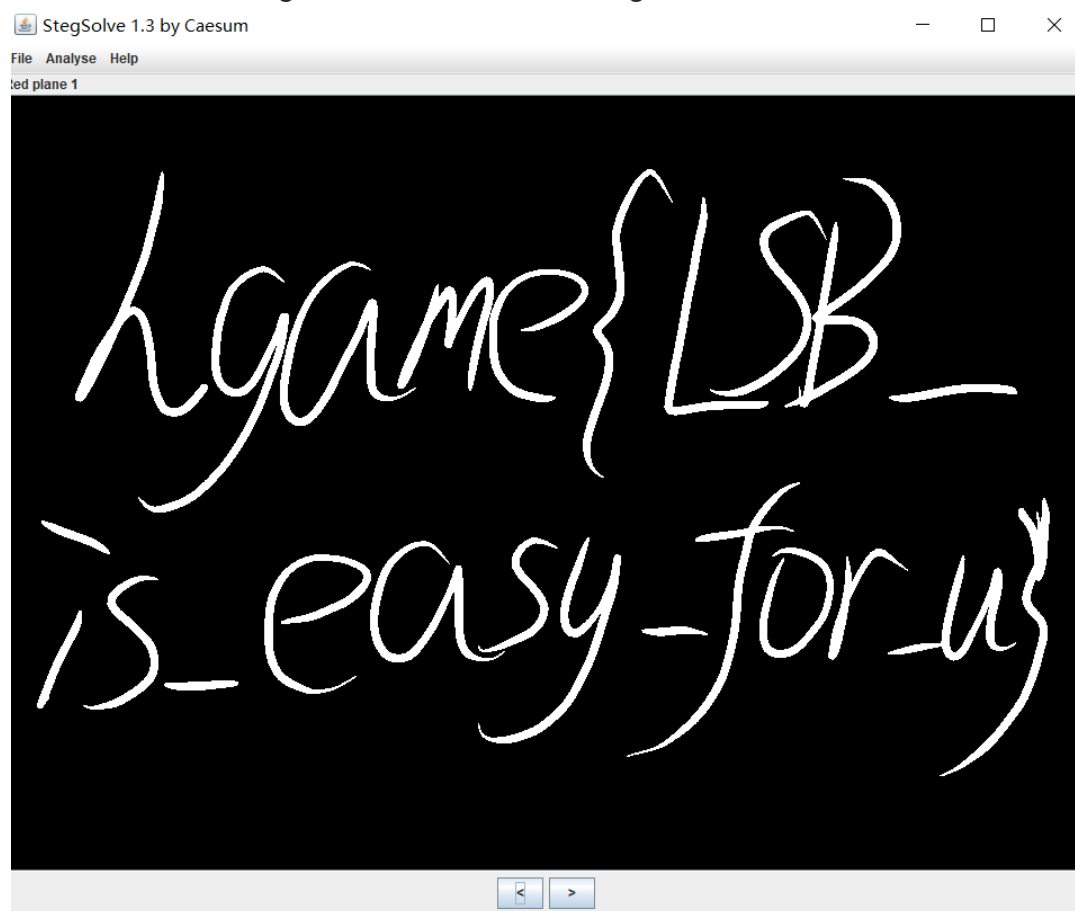
剩下的题目我点击链接下载拖ida里去，蓝的绿的字体颜色晃得我睁不开眼，我不甘心肝了半夜，这才隐隐约约读明白：“re，是不可能re的。代码不会

读，软件又用不来，只有做做签到题才能维持的了分数这样子。往后估计签到题都做不出了。”

pwn同理，但不用往后。

misc1——level

图片下载下来，stegsolve里很简单就找到flag了



misc3——Broken Chest

读题很重要，疯狂钻石查了一下好像是动漫的梗，和修复有关系。目测是修复压缩包

果然是个损坏的压缩包，但是用winrar自带的功能修复发现修复之后东西没了。

然后winhex打开，因为之前有看过常见文件格式头和尾的特征码总结。于是一眼就发现文件头特征码不对，zip的文件头特征码应该是504B0304，但是这却是4F4B0304，于是就把4F改成50。

压缩包是正常了但是发现flag.txt是加密的，一开始想的是伪加密，就直接把这个压缩包拖到linux的虚拟机，发现并不是，仔细研究发现WinRAR打开这个加密压缩包后右边备注了一串奇怪的字符串，试了试就是压缩包密码。

misc4——无字天书

做完表示叫**我老婆的小秘密**可能更切题

先说下走过的弯路

一开始是个流量包，下载下来用wireshark打开，分组字节流按字符串搜索hgame，发现搜到了，高高兴兴地提取出来发现只是个名为password的txt，内容是hgame*****。因为刚做掉misc3受文件头的影响于是改成按16进制去搜索常见文件格式的文件头特征码，发现貌似有个jpg文件。然后费劲千辛万苦找到头尾然后复制弄出来，结果发现是，居然是deepin的图标。典型的一顿操作猛如虎，一看结果250。

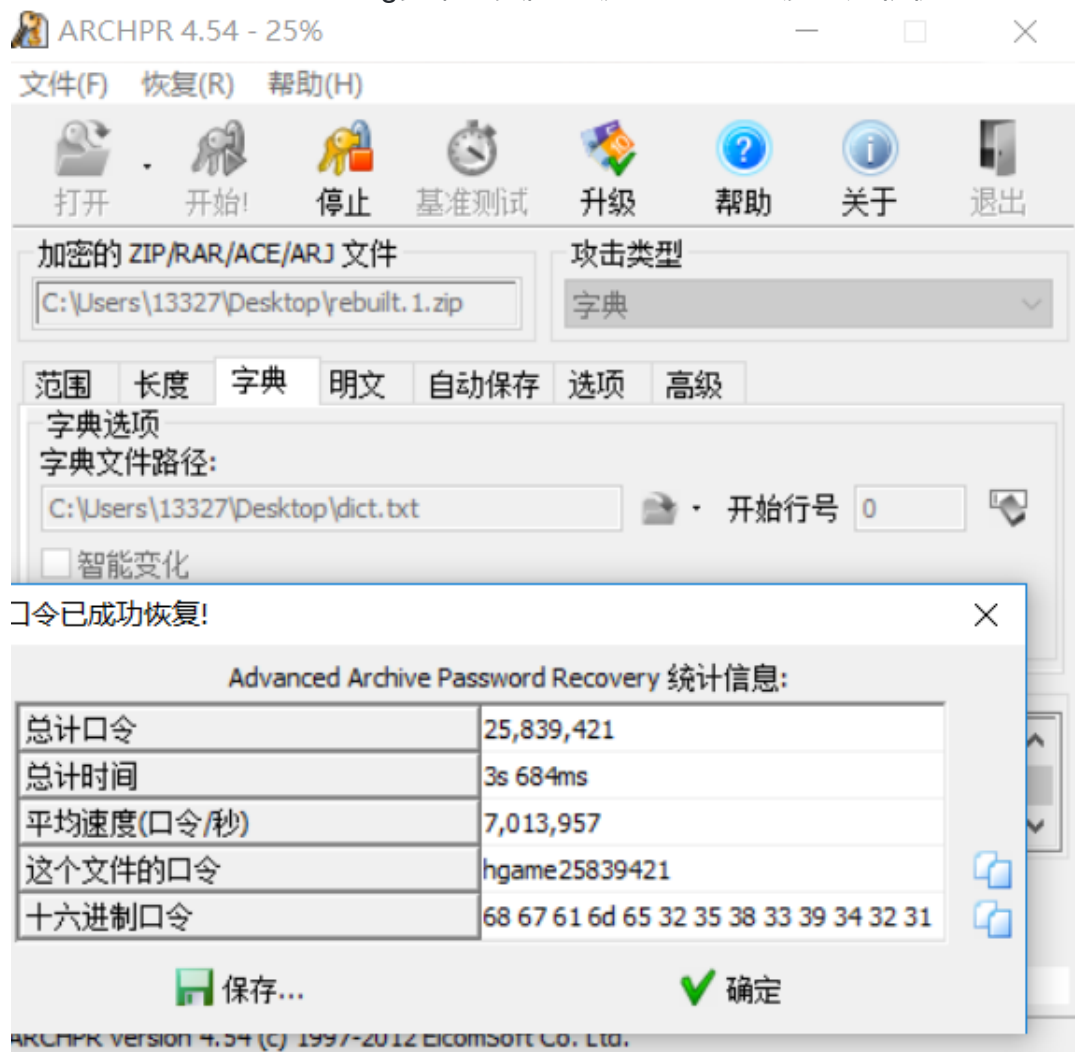
后来重新回到内容囊括hgame这个字符串的数据包，追踪流，搜索504B0304，发现有好多处，我就取了和最近的50 4B相隔最远且16进制转成ascii包含hgame的那个提取出了压缩包。

但是这样提取出来的压缩包是损坏的，用winrar自带的压缩包修复功能修复，发现还是加密的。有张图，然后还有个password.txt。内容即hgame*****试了试发现并不是密码。过了蛮久才来的灵感，*****就像是正常输密码时让密码不可见的处理。但是由于英文字母分大小写26个，0-9，然后特殊字符。这样的组合生成的八位字典实在太恐怖，于是就想了想试试拿0-9去填充后八位，好在成功了。

因为密码是两部分，所以方便起见自己写了个生成字典的小脚本。

```
#coding=utf-8
first="hgame"
txt=open('dict.txt','a')
writedata =lambda data: txt.write(data)
for i in range(1,99999999):
    second=str(i).zfill(8)
    psw=first+second
    writedata(psw)
    writedata('\n')
txt.close()
```

这样生成的字典大小有1个g多，但实际爆破压缩包时速度还是很快的



提取出来的图片发现是可爱的石原小姐姐啊



仔细看了属性没有什么，就kali下用binwalk发现有隐写，是图种。dd命令提取压缩包后，这个加密压缩包是伪加密修复就可以看，是一个空白的word文档，设置中显示隐藏文字就出结果了。