有关思考题的解析和相关内容说明

读读源码

```php
<?php
include ("flag.php");
highlight_file(__FILE__);
echo "<br><br>";

//$flag = 'xxxxxxxx';
$msg_giveme = 'Give me the flag!';
$msg_getout = 'No this. Get out!';
if(!isset($_GET['flag']) && !isset($_POST['flag'])){
    exit($msg_giveme);
}
if($_POST['flag'] === 'flag' || $_GET['flag'] === 'flag'){
    exit($msg_getout);
}
foreach ($_POST as $key => $value) {
    $$key = $value;
}
foreach ($_GET as $key => $value) {
    $$key = $$value;
}
echo 'the flag is : ' . $flag;
?>
```

这里我建立docker文件时,对源码进行了修改,所以这里 `flag` 变量不是在本文件下的 `$flag` ,而是包含进来的 `flag` 变量,而这里意图很明显了,我们需要想办法把包含进来的 `flag` 变量读出来,并且思路也很明显,你看见 `$$` 都在,就是变量覆盖的标志了;

分析分析

- 第一个部分

```php
if(!isset($_GET['flag']) && !isset($_POST['flag'])){
    exit($msg_giveme);
}
```

用 `isset` 判断是否在传入的 `$_POST` 和 `$_GET` 这种全局变量中,是否存在 `flag` 变量,如果不存在,则结束程序的运行,并打印提前定义好的字符.

- 第二部分

```php
if($_POST['flag'] === 'flag' || $_GET['flag'] === 'flag'){
    exit($msg_getout);
}
```

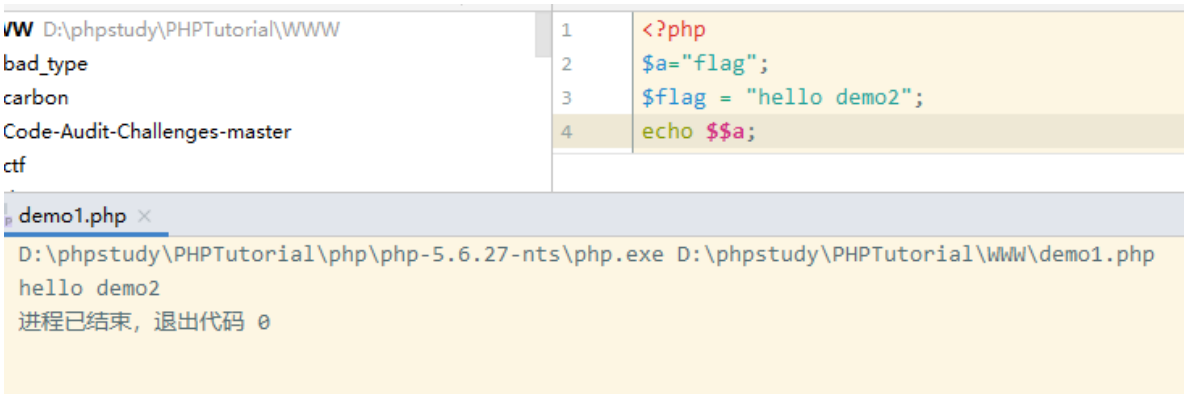
判断我们传入的全局数组中 `flag` 变量的值是否为 `flag` ,有则直接退出

- 第三个部分

这里我就一个循环一个循环的讲吧,第一个循环源码为:

```
foreach ($_POST as $key => $value) {
    $$key = $value;
}
```

这里循环首先将全局数组中的键值对赋值为`$key`和`$value`进行迭代,这里`$key`和`$value`,有个映射关系,比如`$_POST['flag']="this is a demo";`那么这时候`$key`为`flag`,这个`$key`对应的`$value`为`"this is a demo"`,然后由于php支持的`$$`语法效果,我就不多说了,招新题的`GLOBALS`变量就是一个很好的例子,当然为了防止有些人还不会,所以我提供下面的例程:



简单理解你可以当作`$$key => ${$key}`,即先取`{}`内的值,再取`{}`对应内容的值,所以实际上这个循环的功能是将`$_POST`中的键值对,注册成为`php`中相应的变量,最终这些键值对会被转换成`键 = 键值`的php变量

具体的更多的相关内容可以参照我博客的`Unset`板块的相关内容

Moctf-小结

- 第四个部分

```
foreach ($_GET as $key => $value) {
    $$key = $$value;
}
```

这个循环也是类似的功能,只是需要注意这里的值多了一个`$`

讲讲payload

其实payload很简单

```
题目所在域名/?a=flag&flag=a
```

也就是GET传入形似上面的数据即可,由于我懒所以我简单讲两点

- 首先如果进入`POST`的对应循环,再加上前面的判断限制,最终的结果只能是覆盖flag的值~~(必须传入以`flag`为键名的变量)~~,因此我们不能进入`POST`的循环.

  `$$key = $value;`假使我们传入的值为`$_POST['flag'] = "hello"`
  `=>`运行之后会变成`$flag = 'hello'`这里`flag`值被覆盖掉了

- 所以只能进入`GET`循环,为什么`GET`循环可以?因为`value`多了一个`$`,即`$$value`,通过多出来的`$`我们可以将`flag`原有的值转接到另一个变量上去,再重新赋值回来

  `GET`传入`a=flag&flag=a`,相当于`foreach`会执行两次循环
  第一次取值为`$_GET['a']='flag'`

  => `$$key = $$value` => `$a = $flag`,这里`flag`的值被转接到`$a`变量了

  第二次取值再将`$a`的值赋给了`$flag`,`flag`的值又被复原了~~(这里和上面一样,我不细讲)~~,因此最后可以直接打印`flag`