攻防世界——warmup

原创

yybzzz 于 2020-03-19 17:55:34 发布 3817 收藏 10

分类专栏： CTF

本文链接：https://blog.csdn.net/yybzzz/article/details/104971608

一

进入页面发现一个大的滑稽脸，查看源代码，发现提示`source.php`

```
<body>
    <!--source.php-->
```

进入该页面，进行代码审计，又发现一个`hint.php`页面，进入发现提示

flag not here, and flag in ffffllllaaaagggg

表明 flag 在这个文件中，且这个文件名暗示要使用四层目录

二

继续审计代码

```
if  (!  empty($_REQUEST['file'])
        &&  is_string($_REQUEST['file'])
        &&  emmm::checkFile($_REQUEST['file'])
)  {
        include  $_REQUEST['file'];
        exit;
}  else  {
        echo  "<br><img  src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\"  />";
}
```


发现满足三个条件，会包含并运行指定文件`file`，此处的`file`可以由我们构造，为切入点：

1. 检查`file`变量是否为空
2. 检查`file`变量是否为字符串
3. 通过自定义的`checkFile`函数来检查

由于我们要构造payload，前两点直接满足，直接查看`checkFile`函数代码：

```
public static function checkFile(&$page)
{
        $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
        if (! isset($page) || !is_string($page)) {
                echo "you can't see it";
                return false;
        }

        if (in_array($page, $whitelist)) {
                return true;
        }

        $_page = mb_substr(
                $page,
                0,
                mb_strpos($page . '?', '?')
        );
        if (in_array($_page, $whitelist)) {
                return true;
        }

        $_page = urldecode($page);
        $_page = mb_substr(
                $_page,
                0,
                mb_strpos($_page . '?', '?')
        );
        if (in_array($_page, $whitelist)) {
                return true;
        }
        echo "you can't see it";
        return false;
}
```


发现包含四个if语句：

1. 第一个 if 语句对变量进行检验，要求`$page`为字符串，否则返回 false
2. 第二个 if 语句判断`$page`是否存在于`$whitelist`数组中，存在则返回 true
3. 第三个 if 语句，截取传进参数中首次出现`?`之前的部分，判断该部分是否存在于`$whitelist`数组中，，存在则返回 true
4. 第四个 if 语句，先对构造的 payload 进行 url 解码，再截取传进参数中首次出现`?`之前的部分，并判断该部分是否存在于`$whitelist`中，存在则返回 true

以上四个满足一个即可返回 true，若均未满足，则返回 false

三

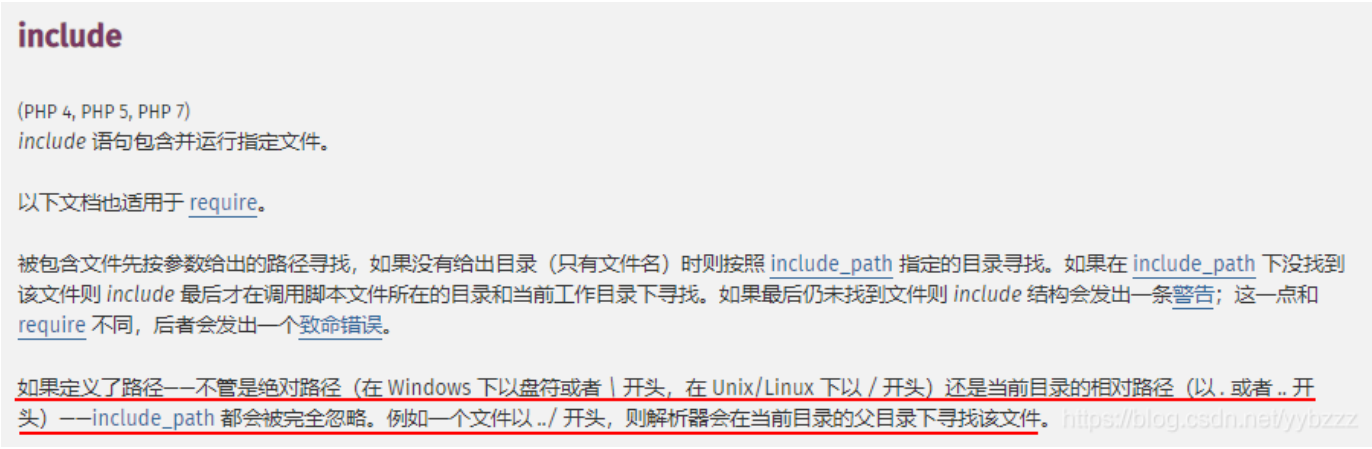
我们利用第三个 if 语句构造参数：

`?file=source.php?/../../../../ffffllllaaaagggg`

第一个`?`表示传参，第二个`?`用来满足截取

四

1. 为什么`include source.php(或hint.php)?/../../../../ffffllllaaaagggg`能执行成功



include

(PHP 4, PHP 5, PHP 7)
include 语句包含并运行指定文件。

以下文档也适用于 require。

被包含文件先按参数给出的路径寻找，如果没有给出目录（只有文件名）时则按照 include_path 指定的目录寻找。如果在 include_path 下没找到该文件则 *include* 最后才在调用脚本文件所在的目录和当前工作目录下寻找。如果最后仍未找到文件则 *include* 结构会发出一条警告；这一点和 require 不同，后者会发出一个致命错误。

如果定义了路径——不管是绝对路径（在 Windows 下以盘符或者 \ 开头，在 Unix/Linux 下以 / 开头）还是当前目录的相对路径（以 . 或者 .. 开头）——include_path 都会被完全忽略。例如一个文件以 ../ 开头，则解析器会在当前目录的父目录下寻找该文件。

因为我们的参数是有`/../../../../`这样的路径，所以符合最后一段话如果定义了路径，就会忽略`/`前的字符串而去找`/../../../../ffffllllaaaagggg`这个文件

2. 网上大多数 writeup 用到的 url 编码绕过发现并没有用到，这一方法的思路是将`?`进行两次 url 编码，变为`%253f`，**在服务器端提取参数时自动解码一次，**`checkFile`函数中解码一次，仍会解码为`?`，可以绕过第四个 if，但是实测只编码一次也可以得到 flag