php加密webshell上传,利用php自包含特性上传webshell

转载

奇舞团 于 2021-03-25 11:30:47 发布 48 收藏
文章标签： php加密webshell上传

0x00 前言

今天做到一题道来自百度杯十二月第四场的ctf题，题目名字叫blog 进阶篇，当时没做出来，看了writeup才知道竟然还有这种骚操作来上传文件进行包含。

0x01 题目复现

前面的解题步骤是注册一个账号以后，在post.php页面提交留言内容，这里会有一个insert into注入，可以拿到管理员的密码。

利用管理员账号登录上去以后会有一个文件包含点，在上级目录下有flag文件。

正常的思路就是使用伪协议来包含flag.php文件，但是这里通过kindeditor编辑器的漏洞遍历文件后尝试包含，会发现包含不了php后缀的文件，其他后缀名的文件可以正常包含

所以这里就要用到一个php上传文件的特性再配合上自包含使得php内存溢出的机制来生成一个webshell文件。

0x01 php文件上传机制

首先先了解一下php的全局数组$_FILES。

官方的解释：

通过 HTTP POST 方式上传到当前脚本的项目的数组。通过使用 PHP 的全局数组 $_FILES，你可以从客户计算机向远程服务器上传文件。

$_FILES 数组提供了多个内容在文件上传时使用，比较重要的有以下几个：

$_FILES['myFile']['name'] 客户端文件的原名称。

$_FILES['myFile']['type'] 文件的 MIME 类型，需要浏览器提供该信息的支持，例如"image/gif"。

$_FILES['myFile']['size'] 已上传文件的大小，单位为字节。

$_FILES['myFile']['tmp_name'] 文件被上传后在服务端储存的临时文件名，一般是系统默认。可以在php.ini的upload_tmp_dir 指定，默认是/tmp目录。

这里的重点就是$_FILES['myFile']['tmp_name']这个变量

上传过程中还利用到了一个重要的函数move_uploaded_file()，该方法是将上传的文件移动到新位置，若不加上这一行代码，临时文件在上传周期后就被删除而不会被存储。

move_uploaded_file(file,newloc)

本函数检查并确保由 file 指定的文件是合法的上传文件(即通过 PHP 的 HTTP POST 上传机制所上传的)。如果文件合法，则将其移动为由 newloc 指定的文件。

0x02 上传测试

在同一目录下创建两个文件，file_upload.html和upload.php

file_upload.html

upload.php

echo "上传前的文件名: ".$_FILES['upload_file']['name'].'>';

echo "上传的临时文件名 : ".$_FILES['upload_file']['tmp_name'].";

echo "文件类型: ".$_FILES['upload_file']['type'].";

echo "文件大小: ".($_FILES['upload_file']['size']/1024).' KB';

echo move_uploaded_file($_FILES['upload_file']['tmp_name'], $_FILES['upload_file']['name']);

?>

上传以后可以看到，tmp_name的命名规则是php[0-9A-Za-z]{3,4}，而且在上传过程中是被临时存储在/tmp目录下(wamp的环境)下。

但是上传完成以后文件会自动被删除,所以在/tmp下找不到这个文件

那么我们要如何做到让阻止他将临时文件删除呢？这里就用到了自包含的特性，让存在php文件包含点的文件包含自己，让他产生一个相当于死循环的状态，在包含的过程中我们进行post文件上传操作。

self_include.php?c=self_include.php

这样就会导致内存溢出，无法正常结束一个php上传周期，这时它会清空自己的内存堆栈，以便从错误中恢复过来，这时对临时文件的删除操作就无法完成，当跳出这个周期后，这个临时文件就以后缀名为tmp的形式保存在/tmp目录下。

这时候我们就利用存在包含点的php文件包含这个临时文件就行了。

0x03 包含测试

测试环境：apache 2.4.9、php版本5.5.12

1. 创建两个文件，一个为存在包含点的self_include.php，一个构造的文件上传点

self_include.php

include($_GET['c']);

?>

self_include.html

File:

2. 我们让他自包含和文件上传同时进行，这里上传一个phpinfo文件。

当我们点击提交以后，发现他报错了

Maximum function nesting level of '100' reached, aborting!

这是因为在我本地装了xdebug插件，它默认只能trace 100条的信息，所以这里在php.ini的xdebug配置下加上一条：

xdebug.max_nesting_level=600

这里测试过大约包含到150次左右程序就会崩溃，就会在tmp目录下生成我们需要的临时文件。

3. 重启服务器，此时重新包含一次，提交

可以看到这里生成了两个临时文件，说明这里经过了两个上传周期，之后php守护进程无法处理这种情况就会抛出一个无法访问异常。

4. 之后就可以直接利用包含点，愉快的包含我们上传的文件了

5. 所以在如果在远程服务器上的php脚本存在文件包含点的话，我们就可以在本地构造一个html文件，action指向他提交过去就行了。

就上面那题来说，最后为了找到文件名，用了kindeditor编辑器的目录遍历漏洞来找到临时文件的文件名

payload: /kindeditor/php/file_manager_json.php?path=../../../../tmp/

在实战中有其他两种方式可以包含到临时文件：

使用爆破的方式找出文件名

最容易爆的是三位数字，所以这里可以多尝试上传几次，直到有三位数字的临时文件生成。

在windows系统下可以使用通配符的方法来包含到临时文件

由于FindFirstFile的特性，在不确定文件后面字符的情况下，可以使用<

http://localhost:9000/upload/self_include.php?c=../tmp/php<<

0x04 脑洞大开

就这题来说还有种包含文件getshell的解法，就是包含日志文件

访问一个不存在的文件时，会在服务器下的/log/access.log进行记录，我们可以通过url写入一个一句话来包含日志文件，从而getshell

1. 首先访问http://localhost:9000/<?php phpinfo();?>，记得这里需要使用bp来发包。

2. 可以看到access.log文件记录下了我们访问的url。

3. 进行文件包含，成功包含了phpinfo文件。

http://localhost:9000/upload/self_include.php?c=../logs/access.log

0x05 总结

这里整个过程需要利用到的点

可控的文件包含点。

目录遍历漏洞。(查看临时文件名)

重新梳理一下思路：

构造一个文件上传点，以post的方式、表单上传(multipart/form-data)的方式上传。

action的url指向存在文件包含漏洞的php文件，接收的参数为自身文件名(self_include.php?c=self_include.php)。

self_include.php进行自文件包含的处理，不断包含自身造成内存溢出。

php守护进程发出内存溢出信号，清空缓冲区和调用堆栈，以便接收新的请求。

一次上传周期未正常结束，/tmp目录下的临时上传文件得以保留。

包含到/tmp目录下的文件，拿到webshell。