

# Apache解析漏洞

---

解析漏洞主要说的是一些特殊文件被IIS、Apache、Nginx在某种情况下被解释成脚本文件格式的漏洞。

## Apache解析漏洞

---

### 漏洞简介

---

Apache文件解析漏洞与用户的配置有密切关系。严格来说，属于用户配置问题，其实apache本身根本不存在所谓的解析漏洞。

### 漏洞原理

Apache默认一个文件可以有多个以点分隔的后缀,当右边的后缀无法识别(不在mime.types内),则继续向左识别。当我们请求这样一个文件:1.php.aaa aaa ->无法识别,向左 php -> 发现后缀是php, 交给php处理这个文件

最后一步虽然交给了php来处理这个文件, 但是php也不认识.aaa的后缀,不解析

其实, 解析漏洞的产生,是由于运维人员在配置服务器时, 为了使apache服务器能解析php, 而自己添加一个handler

```
AddType application/x-httpd-php .php
```

它的作用也是为了让apache把php文件交给php\_module解析,但是注意到它与SetHandler:它的后缀不是用正则去匹配的。所以,在文件名的任何位置匹配到php后缀,它都会让php\_module解析。

Manager	
IPv6 Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, data, http, ftp, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed

This program makes use of the Zend Scripting Language Engine:  
Zend Engine v2.2.0, Copyright (c) 1998-2010 Zend Technologies  
with Xdebug v2.2.7, Copyright (c) 2002-2015, by Derick Rethans



## PHP Credits

## Configuration

### PHP Core

Directive	Local Value	Master Value
allow_call_time_pass_reference	Off	Off
allow_url_fopen	On	On
allow_url_include	On	On
always_populate_raw_post_data	Off	Off

## 漏洞修复

不要使用AddHandler,改用SetHandler,写好正则,就不会有解析问题,

1、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
</FilesMatch>
```

2、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);
```

## Apache 解析漏洞 (CVE-2017-15715)

### 漏洞介绍

Apache HTTPD是一款HTTP服务器,它可以通过mod\_php来运行PHP网页。其2.4.0~2.4.29版本中存在一个解析漏洞,在解析PHP时,1.php\x0a将被按照PHP后缀进行解析,导致绕过一些服务器的安全策略。

### 漏洞原理

1、apache这次解析漏洞的根本原因就是这个 `$`, 正则表达式中,我们都知道\$用来匹配字符串结尾位置

\$符号：匹配输入字符串的结尾位置。如果设置了 `RegExp` 对象的 `Multiline` 属性，则 `$` 也匹配 `'\n'` 或 `'\r'`。  
要匹配 `$` 字符本身，请使用 `\$`。  
如果设置 `MULTILINE` 标示，就当作换符处理，如果不设置就当作一行文本处理

2、本次实验是vulhub下的，所以找到apache配置文件，路径在 `/etc/apache2/` 下，`apache2.conf` 是apache核心配置文件查看其文件发现如下代码：意思是包含这两个文件下的以conf结尾的文件

```
IncludeOptional conf-enabled/*.conf
IncludeOptional sites-enabled/*.conf
```

3、跟进该文件发现关于php的配置文件为 `docker-php.conf`, 打开为如下：

```
<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>
```

4、可以看到在正则中是 `\.php$`, 因为结尾有个 `$` 符号，结合上述原理，`$` 匹配 `'\n'` 或 `'\r'`，所以我们修改数据包在文件名后加 `\n`，`\n` 的十六进制为 `0a`

5、首先查看 `index.php` 文件中的php源码

```
<?php
if(isset($_FILES['file'])) {
    $name = basename($_POST['name']);
    $ext = pathinfo($name,PATHINFO_EXTENSION);
    if(in_array($ext, ['php', 'php3', 'php4', 'php5', 'phtml', 'pht'])) {
        exit('bad file');
    }
    move_uploaded_file($_FILES['file']['tmp_name'], './' . $name);
} else {
?>
```

分析代码文件名 `$name` 是接受POST请求中的数据，为什么要用POST接收呢？因为POST接收不会去掉我们添加的 `\n`, 如果使用 `$_FILES['file']['name']` 去接收文件名，则会吧 `\n` 去掉，所以要满足此漏洞的条件之一就是使用POST接收文件名。

`$ext` 等于POST中的文件名后缀，所以文件名不能是 `['php', 'php3', 'php4', 'php5', 'phtml', 'pht']`，但是可以是 `php\n` 的这种方式，就绕过了if判断。同时这样的文件还以被解析成PHP(为什么能解析成PHP？请看上面的分析)

6、查看 `index.php` 中的html代码

```

<!DOCTYPE html>
<html>
<head><title>Upload</title>
</head>
<body>
<form method="POST" enctype="multipart/form-data">
  <p><label>file:<input type="file" name="file"></label></p>
  <p><label>filename:<input type="text" name="name" value="evil.php"></label></p>
  <input type="submit">
</form>
</body>
</html>

```

html代码中没有什么好说的。

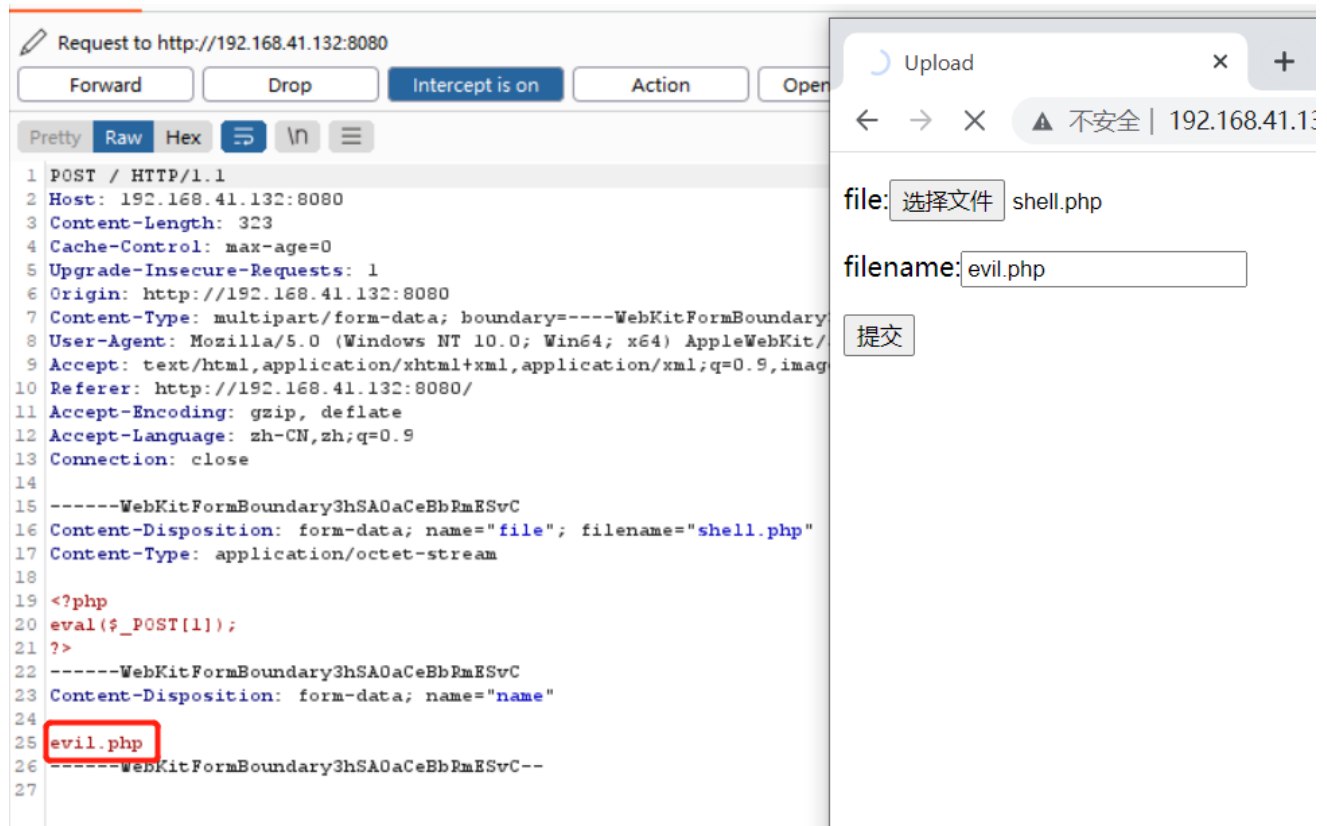
## 7、关于windows和Linux下这个利用是有区别的

linux: 可以正常利用

windows: windows会产生warning的警告, 因为涉及到文件读写, 而windows操作系统不允许后缀以换行符结尾的文件命名方式, 所以这里会文件会创建失败, 就出现了这两个warning了, 产生warning就证明已将绕过了。

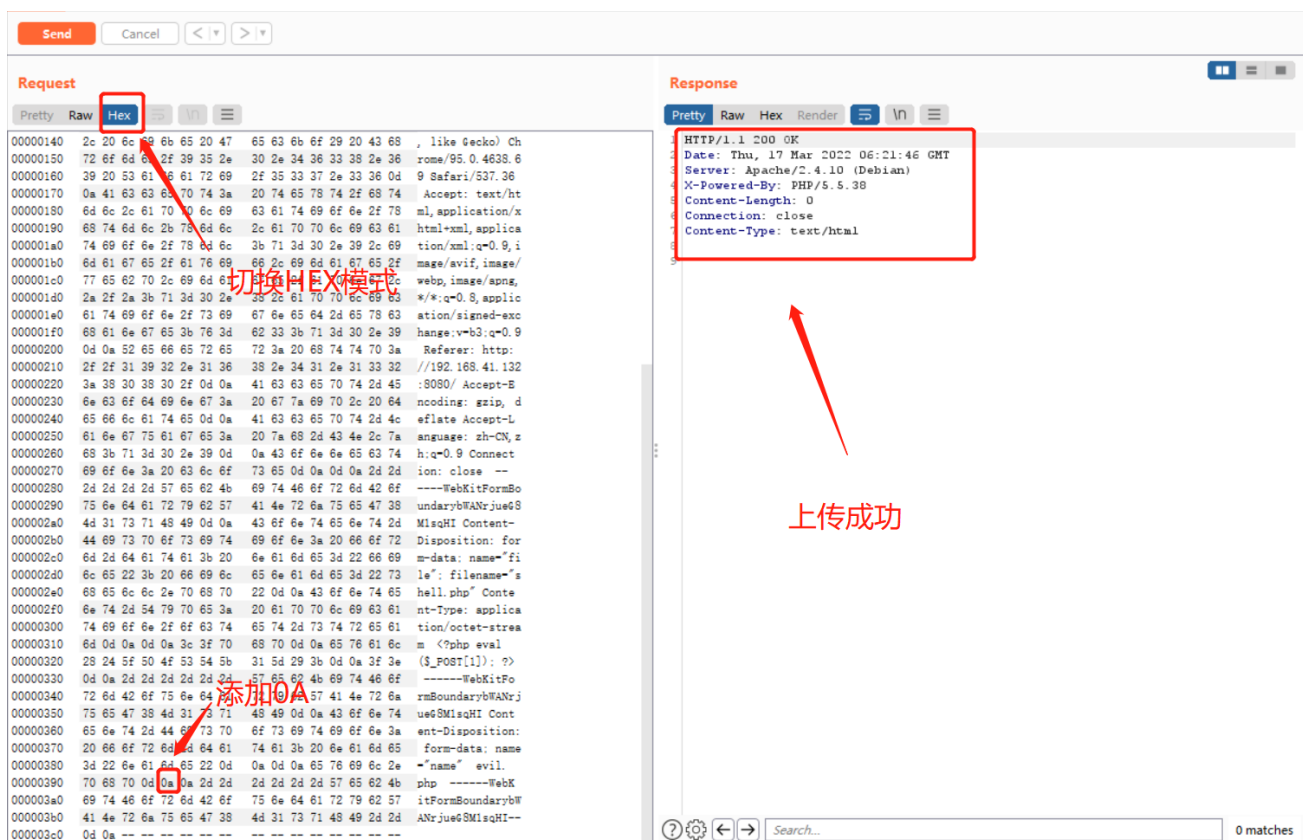
## 漏洞复现

1、首先访问漏洞页面, 并且进行burp抓包, 可以看到POST请求中name为evil.php

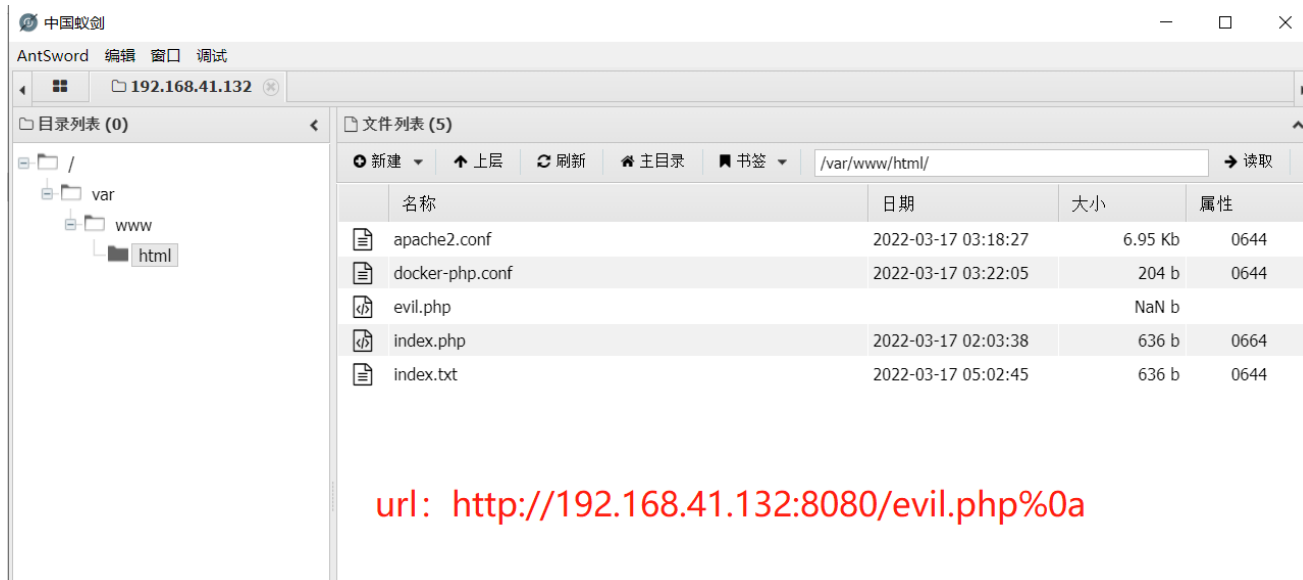


The screenshot shows a web browser window with a form titled "Upload". The form has two fields: "file:" with a "选择文件" (Select File) button and "shell.php" text, and "filename:" with a text input containing "evil.php". A "提交" (Submit) button is at the bottom. To the left, Burp Suite's raw view of the POST request is shown. The request body contains two parts separated by a boundary. The first part is a file upload with filename "shell.php". The second part is a form field with name "name" and value "evil.php", which is highlighted with a red box in the image.

2、将数据包进行修改在evil.php后面添加\n也就是十六进制0A



### 3、访问webshell使用工具连接



## 漏洞修复

- 1、升级到最新版本
- 2、或将上传的文件重命名为为时间戳+随机数+.jpg的格式并禁用上传文件目录执行脚本权限
- 3、获取文件名时用“去掉换行”的函数，比如 `$_FILES['file']['name']`
- 4、在httpd.conf中加入其他正则表达式。