



# 第七讲 文件包含攻击

陈伟

Email: [chenwei@njupt.edu.cn](mailto:chenwei@njupt.edu.cn)

Tel: 18951896489



# 课程内容

1

漏洞原理

2

漏洞利用方式

3

防护手段

4

命令执行攻击







# 什么是文件包含攻击

- ❖ 为了实现单一文件在不同页面的重复使用，通常利用文件包含的方式。
- ❖ 如果某个页面具有这种功能，被包含的文件名（后端代码中include的）可以通过参数的方式被客户端控制，那么存在文件包含漏洞
- ❖ 文件包含漏洞是指当PHP函数引入文件时，没有合理校验传入的文件名，从而操作预想之外的文件，导致文件泄露甚至恶意的代码注入。
  - 本地文件包含（LFI）
  - 远程文件包含（RFI）



```
func.php x one.php two.php
1 <?php
2 function to_print($name){
3     echo "{$name} is print this func<hr/>";
4 }
5 ?>
```

```
func.php one.php x two.php
1 <?php
2 include('func.php');
3 to_print('jack');
```

```
func.php one.php two.php x
1 <?php
2 include('func.php');
3 to_print('bob');
```

127.0.0.1/vul\_code/File

← → ↻ 🏠

jack is print this func

127.0.0.1/vul\_code/File

← → ↻ 🏠

bob is print this func





# 漏洞原理

❖ 严格来说，文件包含漏洞是代码注入的一种

❖ 常见利用方式

- 1 上传木马
- 2 使用包含函数打开木马
- 3 导致木马被执行

❖ 满足两个条件

- 1 采用include（）等文件包含函数
- 2 能够控制包含文件的参数





# PHP文件包含函数



❖ include与require基本是相同的，除了错误处理方面：

- include()在包含的过程中如果出现错误，只生成警告（E\_WARNING），并且脚本会继续
- require()在包含的过程中如果出现错误，会生成致命错误（E\_COMPILE\_ERROR）并停止脚本
- include\_once()与require\_once()，如果文件已包含，则不会包含，以避免函数重定义，变量重新赋值等问题，其他特性如上







# 服务器端功能实现代码

- ❖ 真实环境中，文件包含的利用比较复杂，需要和业务功能相结合。这里使用简单的包含效果来讲解漏洞原理
- ❖ 如果提交的file参数是  
../phpinfo.php
- ❖ 服务器就会执行  
include(../phpinfo.php)

```
<? php
    $file = $_GET["file"];
    if($file) {
        include($file);
    }
?>
```





# 漏洞利用方式1

## ❖ 上传文件包含

- 如果上传的文件包含PHP代码，但是无法执行（无法访问），可以利用包含漏洞上传执行

## ❖ 日志文件包含

- 可以向Web日志中插入PHP代码
- access.log和error.log可能过大





# 漏洞利用方式2

## ❖ 敏感文件包含

- Windows

c:\boot.ini

c:\windows\repair\sam

- Linux

/etc/passwd

/etc/my.conf

/etc/httpd/conf/httpd.conf





# 漏洞利用方式3

## ❖ PHP封装文件协议包含

## ❖ 读取文件

- `php://filter/read=convert.base64-encode/resource=../.. /1.txt` 访问, 可以看到显示了base64编码的内容

## ❖ 写入php文件

- `php://input`

`php://filter` 可以作为一个中间流来处理其他流。

名称	描述
<code>resource=&lt;要过滤的数据流&gt;</code>	这个参数是必须的。它指定了你要筛选过滤的数据流。
<code>read=&lt;读链的筛选列表&gt;</code>	该参数可选。可以设定一个或多个过滤器名称, 以管道符 (   ) 分隔。
<code>write=&lt;写链的筛选列表&gt;</code>	该参数可选。可以设定一个或多个过滤器名称, 以管道符 (   ) 分隔。
<code>&lt;; 两个链的筛选列表&gt;</code>	任何没有以 <code>read=</code> 或 <code>write=</code> 作前缀 的筛选器列表会视情况应用于读或写链。

读取one.php内。容  
`file_get_contents()` 函数是  
用于将文件的内容读入到  
一个字符串中的首选方法

test.php

```
1 <?php
2 echo file_get_contents($_GET['file']);
3 ?>
```

`http://localhost/test.php?file=php://filter/read=convert.base64-encode/resource=one.php`





# 防护手段

## ❖ 漏洞在攻击时主要关注两个点

- 包含目标文件内容的合法性
  - 在文件上传和接口上做好防护
- 包含文件的路径
  - 隐藏路径

## ❖ 防护手段

- 目标的参数过滤
- 中间件级的安全配置





# 防护手段1：文件名验证

❖ 类似防范文件上传攻击

❖ 设置黑白名单

```
case 'jpg'
```

```
case 'png'
```

❖ 文件后缀名固定

```
Include ( "" . $_GET[ 'page' ]. "html" ) ;
```

❖ 绕过方式

- 绕过后缀名 `../../../../boot.ini%00.jpg`
- 目录长度限制 Windows下利用256位截断

%00 截断在 GET 中被 url 解码之后是空字符，字符串的结束符号。



## 防护手段2：路径限制

❖ 针对包含文件的目录进行合法性校验

❖ 目录限制

- 在用户提交的变量前增加固定路径
- Include ‘/var/www/html’ . \$file

❖ 目录回退符过滤

- 避免回退符导致路径变化
- str\_replace( “.” , “ ” , \$str)
- str\_replace( “/” , “ ” , \$str)
- str\_replace( “\\” , “ ” , \$str)

❖ 绕过方式：非常少见，个别利用特殊符号 ~ 回主目录





# 防护手段3：中间件安全配置

## ❖ 调整中间件的安全选项

- 例如Apache中间件+PHP

## ❖ magic\_quote\_gpc

- 对post、get、cookie过来的单引号、双引号等字符增加转义符“\”
- 与SQL注入中的转义方法类似

## ❖ 限制访问区域

- open\_basedir 用来将用户访问文件的活动范围限制在制定区域

## ❖ 设置访问权限







# 命令执行漏洞

- ❖ 类似包含攻击，由于输入的参数被当成命令来执行
- ❖ 远程命令执行漏洞
  - 远程目标站点过滤不严格，将POST的参数以PHP方式执行
  - `@eval($_POST['HITWH']);`
- ❖ 系统命令执行漏洞
  - 利用系统自身的命令实现额外的命令
  - `shell_exec('ping' . $target)`
- ❖ 远程命令执行漏洞威胁远大于本地命令执行漏洞
  - Struts2远程执行漏洞已经造成非常大的影响





# 远程命令执行漏洞

## ❖ 利用系统函数实现远程命令执行

- `eval()`
  - 语法严格，会报错
- `assert()`
  - 不严格
- `preg_replace()`
  - 早期版本如果存在“/e”修饰符，后面的值会被当成PHP执行
- `array_map()`, `call_user_func()` .....





# 系统命令执行漏洞

## ❖ 利用PHP的系统命令执行函数来调用系统命令并执行

- `system()`
- `exec()`
- `shell_exec()`
- `passthru()`

## ❖ DVWA为例

- `shell_exec('ping '.$target);`
- 输入命令 `8.8.8.8&&ipconfig`, 利用&&进行连接







# 防护方案

## ❖ 禁用部分系统函数

- `phpinfo()`、`eval()`、`exec()`、`system()`

## ❖ 严格过滤关键字符

- `&&` ; `||`

## ❖ 严格限制运行的参数类型

- 例如对于IP地址进行限制

