文件系统安全

PHP 遵从大多数服务器系统中关于文件和目录权限的安全机制。

这就使管理员可以控制哪些文件在文件系统内是可读的。必须特别注意的是全局的可读文件，并确保每一个有权限的用户对这些文件的读取动作都是安全的。

PHP 被设计为以用户级别来访问文件系统，所以完全有可能通过编写一段 PHP 代码来读取系统文件如 /etc/passwd，更改网络连接以及发送大量打印任务等等。因此必须确保 PHP 代码读取和写入的是合适的文件。

请看下面的代码，用户想要删除自己主目录中的一个文件。假设此情形是通过 web 界面来管理文件系统，因此 Apache 用户有权删除用户目录下的文件。

示例 #1 不对变量进行安全检查会导致

```
<?php
// 从用户目录中删除指定的文件
$username = $_POST['user_submitted_name'];
$userfile = $_POST['user_submitted_filename'];
$homedir = "/home/$username";
unlink ("$homedir/$userfile"); // 删除文件
echo "The file has been deleted!";
?>

1.root
2.不可以，直接使用用户输入来删除文件；
3.cd ../../../../../../../etc/passwd

1.服务本身的权限控制；
2.用户权限控制；
  1.敏感文件只有管理员有权限；
  2.用户只能操作自身的文件；
```

既然 username 和 filename 变量可以通过用户表单来提交，那就可以提交别人的用户名和文件名，甚至可以删掉本来不应该让他们删除的文件。这种情况下，就要考虑其它方式的认证。想想看如果提交的变量是“../etc/”和“passwd”会发生什么。

示例 #2文件系统攻击

```php
<?php
// 删除硬盘中任何 PHP 有访问权限的文件。如果 PHP 有 root 权限:
$username = $_POST['user_submitted_name']; // "../etc"
$userfile = $_POST['user_submitted_filename']; // "passwd"
$homedir  = "/home/$username"; // "/home/../etc"

unlink("$homedir/$userfile"); // "/home/../etc/passwd"

echo "The file has been deleted!";
?>
```

有两个重要措施来防止此类问题。

- 只给 PHP 的 web 用户很有限的权限。
- 检查所有提交上来的变量。