

# 第 5 章

# PHP 安全编码

随着互联网的发展,Web 2.0、云计算、物联网等热门概念不断催生新的产业和服务,与此同时,PHP 作为支撑这些新的产业和服务的技术体系也得到了空前发展。本章首先概述 PHP 开发的安全现状;其次对常见的 PHP 安全漏洞进行原理分析,提出可用的防范方案;最后总结 PHP 安全编码规范,以帮助程序设计人员更好、更安全地进行 Web 开发。

## 5.1

## PHP 开发安全现状

互联网快速发展、不断创新的特点使得网站开发必须以最快的开发速度和最低的成本完成,才能保持领先性,以吸引更多的网民。而 PHP 的运行效率高、开发速度快、可扩展性强、开源自由等优点非常符合目前的互联网发展趋势,因此,越来越多的 Web 应用选择 PHP 作为主流的技术方案。

### 1. PHP 的优点

PHP(Hypertext Preprocessor,超文本预处理器)是一种通用开源脚本语言。PHP 最初的英文全称是 Personal Home Page(个人主页),它是用 Perl 语言编写的,是用来显示个人履历以及统计网页流量的个人网页工具程序。后来,随着信息技术的发展,PHP 可以和数据库连接,生成动态网页程序,因此逐步应用于 Web 开发领域,在各种规模的网站中应用广泛。

与其他同类编程语言相比,PHP 有以下优点:

(1) 语法简单。其语法吸收了 Java、C 以及 Perl 等语言的优点,语法简单,初学者容易掌握。

(2) 开源自由。PHP 是免费的开源代码。它有很多较为成熟的资源,例如开源论坛 Discuz!、Phpwind 等,开源框架 Zend Framework、CakePHP、CodeIgniter、symfony 等,开源博客 WordPress 等,开源网店系统 Ecshop、ShopEx 等,开源 SNS 系统 UCHome、ThinkSNS 等,可充分满足使用者的各种应用需求。

(3) 效率高。由于 PHP 是将程序嵌入 HTML 文档中执行的,用 PHP 作出的动态页面的执行效率比完全生成 HTML 标记的 CGI 要高许多,而且 PHP 能实现 CGI 的所有功能。PHP 与一般的脚本代码不同,它可以执行编译后的代码,编译时可以对代码进行加密和优化,使代码运行更快、效率更高。

(4) 可扩展性强。PHP 可以用 C、C++ 等语言进行程序的扩展。

(5) 跨平台能力强。PHP 的跨平台能力使它支持几乎所有流行的数据库以及操作系统。PHP 常与免费的开源 Web 服务器 Apache 和数据库 MySQL 配合,运行在 Linux 平台上(这个组合简称 LAMP,号称“Web 架构黄金组合”),具有很高的性价比,不仅能够降低使用成本,还能够提升开发速度,满足最新的互动式网络应用开发的需求。

(6) 面向对象。PHP 4、PHP 5 在面向对象方面进行了很大的改进,现阶段的 PHP 已经可以用来开发大型商业程序。

PHP 适合快速开发中小型应用系统,能够对变动的需求做出快速反应。目前,在全球几千万个互联网网站中,有半数以上使用的是 PHP 技术。越来越多的公司对开发语言的选择从 ASP、Java 转到了 PHP,这种现象使得 PHP 社区越来越活跃,而活跃的 PHP 社区又反过来影响到很多项目或公司的开发语言选择,形成良性循环。PHP 的快速、开发成本低、周期短、后期维护费用低、开源产品丰富等优点都是其他语言所无法相比的。

## 2. PHP 开发的安全问题

随着越来越多的公司选择使用 PHP 进行网站等 Web 应用程序开发,PHP 开发的安全问题开始慢慢浮现。相对于其他语言,PHP 最大的问题是语法不够严谨,缺乏统一的编码规范。例如,PHP 对函数名、方法名、类名是不区分大小写的,变量不需要定义就可以使用,虽然这些问题在编译时一般可以通过,但是在实际执行中容易发生不可控的问题。攻击者还可能利用这些编码漏洞进行注入攻击,窃取隐私信息,造成系统受损。最初 PHP 采用的是面向过程式编程,因此不同开发者的编程风格各异。虽然现在 PHP 已经支持面向对象,各类框架的命名规范也在一定程度上约束了编码人员的风格,但编码不规范的情况依然非常普遍,造成程序后期维护困难。而其他语言(如 Java)更为规范、严格,有利于维护和阅读,即使是初学者也可以写出规范的代码。

### 1) 弱数据类型语言

PHP 是弱数据类型语言,也称为弱类型定义语言。与强类型定义语言相反,在 PHP 中,数据类型可以被忽略,即在定义一个变量的时候不需要为它指定数据类型,而是在解释的时候根据值的情况动态地赋予变量数据类型。弱数据类型语言不符合“所见即所得”的原则,定义的变量类型是不可预见并且可以改变的,变量类型的不可控性会导致在执行过程中出现大量的变量类型“隐形转换”。在开发人员不清楚“隐形转换”规则的情况下,极容易产生不可预知的运行结果,例如,难以通过编译发现代码缺陷,难以优化编译以提升代码性能,开发时难以做出正确的语言提示,以及容易出现注入漏洞等安全性问题。

### 2) 异常捕获系统能力弱

PHP 的生命周期设计决定了它的异常处理功能使用得并不广泛,因而此功能也一直不够完善。Discuz!、DECMS、Ecshop 等系统的异常处理能力都很弱,一旦执行期间发生异常,就可能导致程序崩溃,这对于业务稳定性要求高的系统影响很大。攻击者只需通过简单的分布式攻击就可以使系统崩溃,影响系统的正常业务。

### 3) 易遭受注入攻击

PHP 文件使用 require() 函数引入或者包含外部文件。当 PHP 文件被执行时,外部文件的内容就将被包含进该 PHP 文件中,如果包含的外部文件发生错误,系统将抛出错

误提示，并且停止 PHP 文件的执行。使用 require() 函数可以包含 URL 或文件名，当函数中存在动态包含文件的时候，攻击者可以利用远程文件使得程序访问并执行恶意文件，会造成一系列安全隐患。

PHP 的安全开发是一个艰巨的任务。以现在的安全技术来说，对于 PHP 安全开发问题的建议是：尽量依据编码规范进行安全编码，以减少可能出现的漏洞；对于数据安全要求高的业务应尽量避免使用 PHP 开发。

## 5.2

# PHP 常见安全漏洞

### 5.2.1 会话攻击

#### 1. 会话攻击的原理

会话攻击是攻击者最常用的攻击手段之一。当客户端的浏览器连接到服务器后，服务器会给该客户端用户生成一个独立的会话(session)，然后交与服务器进行管理和维护，相当于每个用户对应一个服务器的身份 ID，称为会话 ID(session ID)。当用户发送 HTTP 请求时，HTTP 头内将包含会话 ID 的值，服务器通过 HTTP 头的会话 ID 值判断用户的请求，这样可以省去在转换到不同页面进行身份验证时重复输入用户名和密码的麻烦。但这种方法在方便用户的同时却增加了很多安全隐患，如果攻击者能够获得某个用户在某个网站的会话 ID 和其他存储在会话内的重要变量，就可以在此会话 ID 的生命周期内使用该用户的权限以达到自己的某种目的。一个会话的生命周期是从用户通过浏览器连接到服务器后开始的，在用户关闭浏览器、从网站注销或者在 20min 内没有进行任何操作后结束。

与会话有关的攻击主要分为两种。

第一种是会话固定攻击(session fixation)。这种攻击方式的核心要点是让合法用户使用攻击者预先设定的会话 ID 来访问被攻击的应用程序，一旦用户的会话 ID 被成功固定，攻击者就可以通过此会话 ID 来冒充合法用户访问应用程序。

会话固定攻击的应用场景如下：

- (1) 攻击者访问网站 <http://www.abc.com>，获取他自己的会话 ID，如 SID=123。
- (2) 攻击者给目标用户发送链接，并带上自己的会话 ID，如 <http://www.abc.com/?SID=123>。
- (3) 目标用户点击了 <http://www.abc.com/?SID=123> 后，像往常一样，输入自己的用户名、密码，登录到网站。
- (4) 由于服务器的会话 ID 不改变，则现在攻击者点击 <http://www.bank.com/?SID=123> 时，他就拥有了目标用户的身份。

第二种常见的会话攻击是会话劫持。攻击者通过各种攻击手段获取用户的会话 ID，然后利用被攻击用户的身份登录相应网站。

攻击者获取用户的会话 ID 的方法有很多，下面是最常用的 3 种方法。第一种方法是

暴力获取,攻击者可以无数次猜测和尝试,直到蒙对会话 ID 为止,但由于一般 ID 都是随机产生的,这种方法耗时耗力,因此很少使用;第二种方法是通过计算获取,需要攻击者进行大量的逻辑分析和运算,因此这种方法也会耗费大量时间和精力;第三种方法是绝大多数攻击者使用的方法,即通过网络截获、安装病毒或者利用跨站脚本攻击获得用户的会话 ID。

## 2. 会话攻击的防范措施

对于会话攻击,通常用以下几种方法进行防范:

- (1) 定期更换会话 ID,可以用 PHP 自带函数实现这一点。
- (2) 更换会话名称。通常情况下会话的默认名称是 PHPSESSID,这个变量一般是在 Cookie 中保存的。如果更改了会话的名称,攻击者将不容易找到会话的存储地点,就可以阻止攻击者的一部分攻击。
- (3) 对透明化的会话 ID 进行关闭处理。透明化是指在 HTTP 请求中没有使用 Cookie 来指定会话 ID 时,会话 ID 使用链接来传递。关闭透明化会话 ID 可以通过修改 PHP.ini 文件中的相关配置来实现。
- (4) 通过 URL 传递隐藏参数。这样,即使攻击者获取了会话数据,但是由于相关参数是隐藏的,攻击者也很难获得会话 ID。
- (5) 设置 HttpOnly。将 Cookie 的 HttpOnly 设置为 true,可以防止客户端脚本访问 Cookie,从而有效地防止 XSS 攻击。

### 5.2.2 命令注入攻击

#### 1. 命令注入攻击原理

命令注入(command injection)漏洞是 PHP 应用程序中常见的脚本漏洞之一,是指攻击者可以通过构造特殊命令字符串的方式将数据提交至 Web 应用程序中,并利用该方式执行外部程序或系统命令以实施攻击,非法获取数据或者网络资源等。命令注入漏洞的存在十分广泛,国内许多著名的 PHP 应用程序,如 Discuz!、dedecms 等均被发现存在命令注入攻击漏洞。攻击者可以通过命令注入攻击漏洞快速获取网站权限,进而实施挂马、钓鱼等恶意攻击,造成巨大的影响和危害。

命令注入攻击最初被称为 Shell 命令注入攻击,是由挪威一名程序员在 1997 年意外发现的,他通过构造命令字符串的方式从一个网站删除了网页,进行了命令注入攻击。使用系统命令是一项危险的操作,尤其在程序设计人员试图使用远程数据来构造要执行的命令时,如果使用了被污染的远程数据,命令注入漏洞就产生了。命令注入漏洞存在的主要原因是程序设计人员在应用 PHP 语言中一些具有命令执行功能的函数时,对用户提交的数据内容没有进行严格的过滤就带入函数中执行而造成的。例如,当攻击者提交的数据内容为向网站目录写入 PHP 文件时,就可以通过该命令注入攻击漏洞写入一个 PHP 后门文件,进而实施进一步的渗透攻击。

在 PHP 中,可以用以下 4 个命令执行函数和一个运算符执行外部程序或函数:

- (1) system()。该函数可以用来执行一个外部的应用程序并将相应的执行结果输

出,函数原型如下:

```
string system(string command, int &return_var)
```

其中,command 是要执行的命令,return\_var 存放执行命令后的状态值。

(2) exec()。该函数可以用来执行一个外部的应用程序,函数原型如下:

```
string exec(string command, array &output, int &return_var)
```

其中,command 是要执行的命令,output 存放命令输出的每一行字符串,return\_var 存放执行命令后的状态值。

(3) passthru()。该函数可以用来执行一个 UNIX 系统命令并显示原始的输出,当 UNIX 系统命令的输出是二进制的数据,并且需要直接返回值给浏览器时,需要使用 passthru() 函数来代替 system() 与 exec() 函数。passthru() 函数原型如下:

```
void passthru(string command, int &return_var)
```

其中,command 是要执行的命令,return\_var 存放执行命令后的状态值。

(4) shell\_exec()。该函数执行 shell 命令并返回输出的字符串,函数原型如下:

```
string shell_exec(string command)
```

其中,command 是要执行的命令。

(5) “~”运算符。与 shell\_exec() 函数的功能相同,执行 shell 命令并返回输出的字符串。

## 2. 命令注入攻击的防范方法

由于目前 PHP 语言广泛应用于 Web 应用程序开发,Web 应用程序设计人员需要了解命令注入攻击漏洞的危害,修补程序中可能存在的被攻击者利用的漏洞,保护网络用户的安全,使网站免受挂马、钓鱼等恶意代码的攻击。通常程序设计人员可以通过以下几种方法防范命令注入攻击:

- (1) 尽量不要执行外部的应用程序或命令。
- (2) 使用自定义函数或函数库实现外部应用程序或命令的功能。
- (3) 在执行 system() 等命令执行函数前,要确定参数内容。
- (4) 使用 escapeshellarg() 函数处理相关参数。该函数会将任何引起参数或命令结束的字符进行转义,如单引号“'”会被转义为“\'”,双引号“””会被转义为“\"”,分号“;”会被转义为“\;”,这样 escapeshellarg() 函数会将参数内容限制在一对单引号或双引号中,转义后的单引号或双引号无法截断当前命令的执行,达到了防范命令注入攻击的目的。

### 5.2.3 客户端脚本注入攻击

#### 1. 客户端脚本注入攻击原理

客户端脚本注入攻击是指攻击者将可执行的脚本插入表单、图片、动画或超链接文字等对象内,当用户留言或者打开这些对象后,浏览器将执行攻击者所植入的恶意脚本,通过使用户自动跳转到一些攻击者恶意构造的网站等行为实施攻击。此漏洞在以前的

PHP 网站中经常存在,但随着 PHP 版本的升级,这些问题已经慢慢减少。

可以用于脚本植入的 HTML 标签一般包括以下几种:

(1) <script> 标签标记的 JavaScript 和 VBScript 等页面脚本程序。在<script>标签内可以指定 JavaScript 程序代码,也可以在 src 属性内指定 JavaScript 文件的 URL 路径。

(2) <object> 标签标记的对象。这些对象是 Java Applet、多媒体文件和 ActiveX 控件等。通常在 data 属性内指定对象的 URL 路径。

(3) <embed> 标签标记的对象。这些对象是多媒体文件,例如 swf 文件。通常在 src 属性内指定对象的 URL 路径。

(4) <applet> 标签标记的对象。这些对象是 Java Applet,通常在 codebase 属性内指定对象的 URL 路径。

(5) <form> 标签标记的对象。通常在 action 属性内指定要处理表单数据的 Web 应用程序的 URL 路径。

客户端脚本注入攻击的步骤如图 5-1 所示。具体如下:

- (1) 攻击者在注册成普通用户后,登录网站。
- (2) 打开留言页面,插入用来进行攻击的 JavaScript 代码。
- (3) 其他用户(包括管理员)登录网站,浏览此留言的内容。
- (4) 隐藏在留言内容中的 JavaScript 代码被执行,攻击被发动。

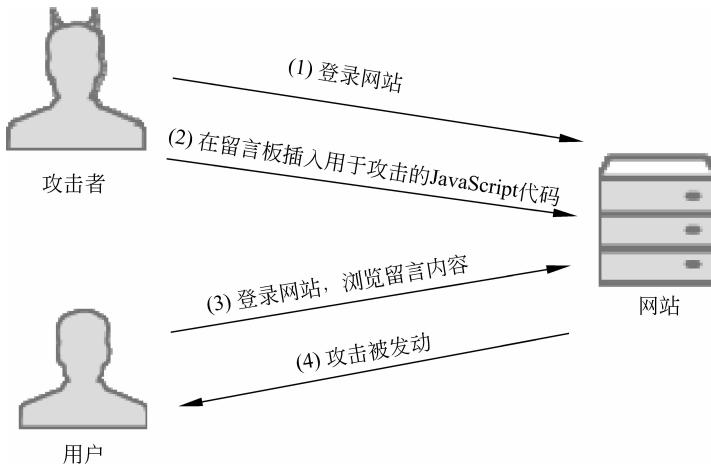


图 5-1 客户端脚本注入攻击步骤

## 2. 客户端脚本注入攻击的防范方法

攻击者利用客户端脚本注入漏洞进行攻击的手段是灵活多变的,因此需要采取多种防范方法,才能有效防止攻击者对脚本注入漏洞进行攻击。常用的方法如下:

(1) 对可执行文件的路径进行预先设定。可以通过设置安全模式 safe\_mode\_exec\_dir 来实现。如果 PHP 使用了安全模式,system()和其他程序执行函数将拒绝启动不在此目录中的程序,必须使用/作为目录分隔符。

(2) 对命令参数进行处理,一般使用 `htmlspecialchars()` 函数来将特殊字符转换成 HTML 编码。

(3) 用系统自带的函数库代替外部命令。

(4) 进行系统操作的时候尽可能不使用外部命令。

### 5.2.4 变量覆盖漏洞

#### 1. 常见的变量覆盖漏洞

变量覆盖漏洞是指用传入的参数值替换程序原有的变量值的情况。一般变量覆盖漏洞需要结合程序的其他功能实现完整的攻击。经常导致变量覆盖漏洞的场景有:全局变量注册未关闭, \$\$ 遍历初始化变量,`extract()`函数、`parse_str()`函数和 `import_request_variables()`函数使用不当,等等。

##### 1) 全局变量漏洞

PHP 中的全局变量不像其他开发语言一样需要预先声明,而是可以直接使用,PHP 会在第一次使用时自动创建变量,并根据上下文环境自动确定变量的类型。这对于程序设计人员而言是十分方便的,因为只要一个变量被创建,就可以在程序中的任何地方使用。但 PHP 的这种特性也导致程序员在编写 PHP 程序的过程中很少初始化变量,通常直接使用变量被创建时默认的空值,这使得攻击者可以通过给全局变量赋值来欺骗代码,达到其恶意的目的。

`register_globals` 是 PHP 中的一个控制选项,其意思是注册为全局变量。当其状态是 `on`(开启)的时候,传递过来的值会被直接注册为全局变量,如果该变量在此之前有值,这个值会被覆盖,因此变量的 `register_globals` 选项应时刻保持 `off`(关闭)状态,以防止攻击者恶意传递参数值,覆盖系统变量。`register_globals` 选项在 PHP 4 及以前版本默认开启,在 PHP 5 以后默认关闭。

##### 2) \$\$遍历初始化变量问题

使用 `foreach` 来遍历数组中的值时,会将获取的数组的键名作为变量,将数组的键值作为变量的值,因此就产生了变量覆盖漏洞。请求 `?id=1` 会将 `$ id` 的值覆盖,使得 `$ id=1`。

##### 3) `extract()` 变量覆盖

`extract()` 函数从数组中将变量导入当前的符号表,使用数组键名作为变量名,使用数组的键值作为变量值,针对数组中的每个元素在当前符号表中创建一个对应的变量。该函数在以下 3 种情况下会覆盖已有变量的值:

(1) 当第二个参数为 `EXTR_OVERWRITE` 时。这表示如果有冲突,则覆盖已有的变量的值。

(2) 当只传入第一个参数时。这时候默认为 `EXTR_OVERWRITE` 模式。

(3) 当第二个参数为 `EXTR_IF_EXISTS` 时。这表示仅在当前符号表中存在同名变量时覆盖它们的值。

##### 4) `parse_str()` 变量覆盖

`parse_str()` 函数的作用是解析字符串并将其注册为变量。`parse_str()` 函数有两个参数:

第一个参数代表要解析并注册为变量的字符串;第二个参数是一个数组,注册的变量会放到这个数组里。如果没有第二个参数,则由该函数设置的变量将覆盖已存在的同名变量。

### 5) import\_request\_variables() 变量覆盖

import\_request\_variables() 函数的作用是把 GET、POST 和 COOKIE 的参数注册为变量,该功能用在 register\_globals 为 off 的时候。不过,这个函数在 PHP 5.4 之后就被取消了。

import\_request\_variables() 函数有两个参数:第一个参数代表要注册的变量,其中 G、P、C 分别代表 GET、POST、COOKIE;第二个参数为要注册的变量前缀。import\_request\_variables() 函数可以在 register\_global 为 off 时将 GET、POST、COOKIE 变量导入全局作用域中。

## 2. 变量覆盖漏洞的防范方法

对于 PHP 的变量覆盖漏洞问题,通常采用以下的防范方法:

(1) 使用原始变量数组。建议直接用原始的变量数组(如 \$\_POST、\$\_GET 等)进行操作,避免使用变量注册功能。

(2) 验证变量是否存在。注册变量前,先判断变量是否存在。使用 extract() 函数时可以配置第二个参数为 EXTR\_SKIP。自行创建的变量一定要初始化。

### 5.2.5 危险函数

为了系统的安全起见,常在程序设计时禁用一些 PHP 危险函数。具体如下:

`phpinfo()`

功能描述:输出 PHP 环境信息以及相关的模块、Web 环境等信息。

危险等级:中。

`passthru()`

功能描述:允许执行一个外部程序并回显输出,类似于 exec()。

危险等级:高。

`exec()`

功能描述:允许执行一个外部程序(如 UNIX Shell 或 CMD 命令等)。

危险等级:高。

`system()`

功能描述:允许执行一个外部程序并回显输出,类似于 passthru()。

危险等级:高。

`chroot()`

功能描述:可改变当前 PHP 进程的工作根目录,仅当系统支持 CLI 模式的 PHP 时才能工作,且该函数不适用于 Windows 系统。

危险等级:高。

`scandir()`

功能描述：列出指定路径中的文件和目录。

危险等级：中。

`chgrp()`

功能描述：改变文件或目录所属的用户组。

危险等级：高。

`chown()`

功能描述：改变文件或目录的所有者。

危险等级：高。

`shell_exec()`

功能描述：通过 shell 执行命令，并将执行结果作为字符串返回。

危险等级：高。

`proc_open()`

功能描述：执行一个命令并打开文件指针用于读取以及写入。

危险等级：高。

`proc_get_status()`

功能描述：获取使用 `proc_open()` 打开的进程的信息。

危险等级：高。

`error_log()`

功能描述：将错误信息发送到指定位置(文件)。

安全备注：在某些版本的 PHP 中，可使用 `error_log()` 绕过 PHP 安全模式，执行任意命令。

危险等级：低。

`ini_alter()`

功能描述：它是 `ini_set()` 函数的一个别名函数，两者功能相同。具体参见 `ini_set()`。

危险等级：高。

`ini_set()`

功能描述：可用于修改、设置 PHP 环境配置参数。

危险等级：高。

`ini_restore()`

功能描述：可用于将 PHP 环境配置参数恢复为初始值。

危险等级：高。

`d1()`

功能描述：在 PHP 运行过程中(而非启动时)加载 PHP 外部模块。

危险等级：高。

`pfssockopen()`

功能描述：建立一个 Internet 或 UNIX 域的 socket 持久连接。

危险等级：高。

`syslog()`

功能描述：可调用 UNIX 系统的系统层 syslog() 函数。

危险等级：中。

`readlink()`

功能描述：返回符号连接指向的目标文件内容。

危险等级：中。

`symlink()`

功能描述：在 UNIX 系统中建立一个符号链接。

危险等级：高。

`popen()`

功能描述：可通过该函数的参数传递一条命令，并执行该函数打开的文件。

危险等级：高。

`stream_socket_server()`

功能描述：建立一个 Internet 或 UNIX 服务器连接。

危险等级：中。

`putenv()`

功能描述：用于在 PHP 运行时改变系统字符集环境。在低于 5.2.6 版本的 PHP 中，可利用该函数修改系统字符集环境，然后利用 sendmail 指令发送特殊参数，执行系统 shell 命令。

危险等级：高。

### 5.3

## PHP 安全编码规范

当一个软件项目尝试制定一致的编码规范时，可以使参与项目的开发人员更容易了解项目中的代码，弄清程序的状况，使新的参与者可以很快地适应环境，防止部分参与者为了节省时间而自创一套风格，导致其他项目人员在阅读程序时浪费过多的时间和精力。