第18章　使用PHP和MySQL实现安全事务

在本章中，我们将解释如何在输入、传输和存储过程中安全地处理用户数据。这将允许我们以端到端的方式，在网站与用户之间进行安全的事务处理。

在本章中，我们将主要介绍以下内容：

- 提供安全事务处理
- 使用Secure Sockets Layer（SSL，加密套接字层）
- 提供安全存储
- 确定是否需要存储信用卡号码
- 在PHP中使用加密技术

18.1　提供安全的事务处理

实际上，在使用Internet的过程中，提供安全的事务处理就是这样的问题：检查系统中信息的流动，确保在每一点的信息都是安全的。在网络安全问题中，没有绝对的安全。没有系统在过去或将来都无法入侵。对于安全来说，我们的意思是保护一个系统或者一次传输所付出的努力与涉及的信息的价值相比较而言的。

如果要有效地在安全方面付出正确的努力，需要在系统各个部分检查信息流。一个典型应用程序的用户信息流如图18-1所示，该程序使用PHP和MySQL编写。

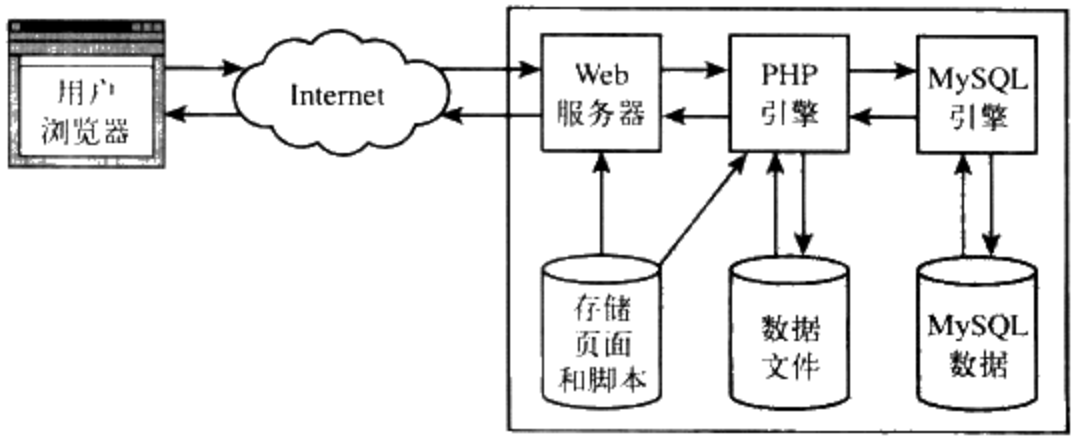

图18-1　一个典型Web应用程序环境的元素将用来存储与处理用户信息

出现在每个系统中的每个事务处理的细节各不相同，这取决于系统设计和用户数据以及触发事务的行为。我们可以用类似的方法检查所有这些细节。Web应用程序与用户之间的每个事务都是以用户使用浏览器经由Internet向Web服务器发送请求为开始的。如果被请求页面是一个PHP脚本，Web服务器会将该页委托给PHP引擎处理。

该PHP脚本可能要读写磁盘数据。它也可以使用`include()`或`require()`来包含其他PHP或HTML文件。它还可以向MySQL后台发送SQL查询并接收响应。MySQL引擎负责读写引

擎所需处理的磁盘数据。

该系统由3个主要部分组成：

- 用户机器
- Internet
- 我们的系统

接下来，我们将分别讨论每个部分的安全问题，显然，在很大程度上，用户机器和Internet的安全将超出控制范围。

18.1.1 用户机器

从我们的角度出发，用户机器正在运行一个Web浏览器。我们无法控制其他一些因素，例如，该机器配置的安全程度。需要记住的是，这台机器可能是非常不安全的，或者甚至是图书馆、学校或咖啡厅中的一台共享终端。

如今，人们可以使用许多不同浏览器，而每个浏览器都有各自不同的功能。如果只考虑目前最流行的两种浏览器最新版本，它们之间的大多数差异只会影响HTML的显示方式，但是，我们还需要考虑一些安全问题或功能问题。

我们应该注意到，如果考虑到安全或隐私风险，一些人将不会使用一些有用的特性，例如，Java、cookies或JavaScript。如果使用这些特性，就必须对一些情况进行测试，例如，对于不使用这些特性的用户来说，是否会降低应用程序性能，或者，考虑提供一种要求特性更少，但界面更丰富的应用程序以允许这些人的访问。

美国和加拿大以外的用户使用的Web浏览器可能只支持40位加密。尽管在2000年1月，美国政府已经对法律进行了修改，允许出口功能强大的加密技术（到非禁止国家），支持128位加密的Web浏览器对大多数用户来说也已经可以使用，但一些旧版本的浏览器仍将不会升级。除非在网站的文本中向用户保证了安全性，否则对于一个Web开发人员来说，它并不会引起重视。SSL将自动使服务器和用户浏览器以它们都知道的最安全级别进行通信。

我们不能保证一个正在连接站点的Web浏览器是否正在通过我们所希望的接口进行交互。

网站的请求可能来自另一个网站，它可能企图窃取我们网站的图片或者内容，或者来自一个使用类似于cURL软件的人，该软件可以成功地绕过安全措施。

cURL软件库可以用于模拟浏览器连接，要了解其详细内容，请参阅第20章。该软件库对开发人员来说非常有意义，但是，它也有可能被怀有恶意的人所使用。

虽然我们不能改变或控制用户的机器配置，但是一定要牢牢记住这点。用户机器的可变性可能是决定我们所提供的功能能够发挥多大作用的一个重要因素，这些功能包括我们通过服务器端脚本（例如，PHP）提供的功能，以及客户端脚本（例如，JavaScript）提供的功能。

PHP提供的功能能够与每个用户的浏览器兼容，因为最终结果通常都是HTML页面。使用的任何东西，除了基本的JavaScript之外，就要考虑不同浏览器版本的兼容性了。

从安全的角度来看，使用服务器端脚本来处理一些事情，例如数据验证，还是比较好的，因为这样，用户就无法看到我们的源代码。如果在JavaScript中验证数据，用户能够看到代码，并且可能绕过这些代码。

需要保留的数据可以作为文件或者数据库记录保存到服务器中，也可以作为cookies保存到用户的机器中。要了解如何使用cookies保存一些有限的数据（例如，会话密钥），请参阅第23章的详细介绍。

我们保存的大部分数据应该在服务器上或者数据库中。当然，也有许多好的理由要将小信息尽可能地保存在用户的机器中。如果信息在系统之外，那么我们将无法控制这些信息存储的安全性，也不能确认用户是否已经删除了它，甚至不能阻止用户修改它，从而使系统产生混淆。

18.1.2 Internet

与无法控制用户机器一样，我们也不能控制Internet的特性。但是，就像用户机器一样，这并不意味着在设计系统的时候可以忽略这些特性。

Internet有许多迷人的特性，但是它与生俱来就是不安全的网络。将信息从一端发送到另一端的时候，必须要记住，别人可能会查看或者修改正在传输的信息，我们已经在第15章讨论了这些问题。考虑到这些因素，我们可以决定采取什么行动。

所采取的行动可能包括：

- 无论如何都要传输信息，这些信息无法保密并且在传输过程可能会被修改。
- 在传输前数字签名该信息，确保不会被修改。
- 在传输前加密信息，确保其保密并且不会被修改。
- 因为信息非常重要而不可冒险被人拦截，决定使用其他方式分发信息。

Internet也是一个匿名的媒体。确认正在与我们联系的人就是他本人也非常困难。即使能够保证对方的身份，要在一些场合如法院上证实这一点也是非困难的。这将引起否认问题，该问题我们已经在第15章中讨论了。

总之，通过Internet进行事务处理的时候，保密与否认是两个重要的问题。

至少有两种方法可以用来保护经过Internet流入和流出Web服务器的信息：

- SSL
- S-HTTP（Secure Hypertext Transfer Protocol，加密的超文本传输协议）

这两种技术都能提供加密和防止修改信息的身份验证功能，但是，SSL是随时可以获得的，并且得到广泛应用，而S-HTTP还没有真正普及。我们将在本章的后续内容中详细讨论SSL。

18.1.3 我们的系统

我们可以控制的领域只有自己的系统。在图18-1中，右边方框内的组件代表我们的系统。

这些组件可能分布在一个网络中，也可能存在于一台物理机器内。

当我们用来分发Web内容的第三方产品能够处理信息的安全时，就不用担心这些信息的安全性，它们是相当安全的。这些特殊软件的编写人员已经尽可能地考虑过了它们的安全性。只要正在使用一个著名产品的最新版本，通过Google或其他搜索引擎，就能够发现任何众所周知的问题。我们应该保持随时掌握这些信息。

如果安装和配置也是任务的一部分，那么就需要考虑软件安装和配置的方式。许多安全问题的出现都是由于没有遵循警告的结果，或者由于常规的系统管理问题引起（这是其他图书的

主题）。买一本关于管理操作系统方面的图书，可以随时参考，或者雇佣一位专家级别的系统管理员都是很好的办法。

在安装PHP时，需要考虑一个特殊的问题，这就是以Web服务器的SAPI模块的方式安装PHP比以CGI接口的方式安装PHP更安全、更有效。

作为一个Web应用程序开发人员，需要考虑的主要问题是我们自己的脚本能够做什么或者不能做什么。通过Internet，应用程序会将什么样的重要数据传递给用户呢？我们要求用户将什么样的重要数据传递给我们自己呢？如果我们正在传输的信息是与用户之间的秘密信息或者要求中间媒介很难修改它们，那么就应该考虑使用SSL。

我们已经讨论过在用户的计算机和服务器之间使用SSL。此外，还应该考虑将数据从系统的一个组件传递到另一个组件的情况。一个典型的例子是MySQL数据库位于与Web服务器不同的机器上。PHP将通过TCP/IP连接MySQL服务器，这种连接是不加密的。如果这些机器都是在私有局域网上，那么需要确保局域网是安全的。如果这些机器是通过Internet进行通信的，系统将可能运行得比较慢，同时需要将这种连接当作基于Internet之上的其他连接一样处理。

当用户认为我们正在与他们进行交互，他们的确是在和我们进行交互，确保这一点是非常重要的。注册一个数字证书可以保护我们的访问者免受网络欺骗（那些冒充是我们网站的人），还可以允许我们在避免用户看到一个警告信息的前提下使用SSL，以及为在线用户提供一种值得信任的氛围。

我们的脚本仔细检查用户输入的数据了吗？我们是否关心了存储信息的安全性？在本章接下来的几节中，我们将逐一回答这些问题。

18.2　使用加密套接字层（SSL）

Secure Sockets Layer Protocol（SSL协议）最初是由Netscape公司提出来的，它是为了实现Web服务器和Web浏览器之间的安全通信而设计的。自从被采纳以来，已经成为浏览器与服务器之间交换重要信息的非正式标准方式。

SSL 2和3版本都得到了广泛的支持。大多数Web服务器都包含SSL功能，或者作为插件模块的方式接受它。Internet Explorer和Firefox都支持SSL 3及其以后的版本。

通常，实现SSL的网络协议和软件都是按照一种层次堆栈的形式来组织的。每一层能够将数据传输到上一层或下一层，并且能够向上一层和下一层发出服务请求。图18-2所示的就是这种堆栈式的协议。

当使用HTTP传输信息时，HTTP协议将调用传输控制协议（TCP）层，而传输控制协议又依赖于Internet协议（IP）层。该协议又需要一个适用于网络硬件的协议。这种协议用于将数据打包并以电子信号方式把它们传输到目的地。

HTTP被应用层协议调用。有许多其他不同的应用层协议，例如FTP、SMTP和Telnet（如图18-2所示），以及其他协议如POP和IMAP。TCP是在TCP/IP网络中使用的两种传输层协议中的一种。IP是网络层的协议。网络层的主机

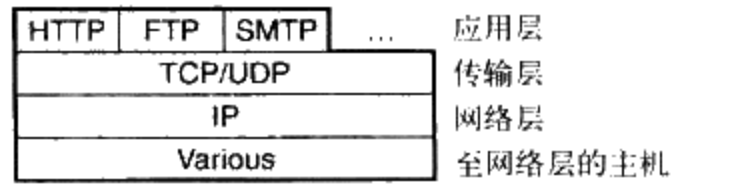

图18-2　应用层协议（例如，超文本传输协议）使用的协议堆栈

负责将我们的主机（计算机）连接到一个网络。TCP/IP协议堆栈没有详细说明应用于这一层的协议，因为对于不同类型的网络需要使用不同的协议。

在发送数据时，数据将通过堆栈从应用层发送到物理层网络媒介。一旦接收到数据，数据将通过堆栈从物理层传输到应用层，直到应用程序。

使用SSL将为这种模型添加一个额外的透明层。SSL层介于传输层和应用层之间，如图18-3所示。在将信息发送到传输层以将其送到目的地之前，SSL层修改来自HTTP应用层的数据。

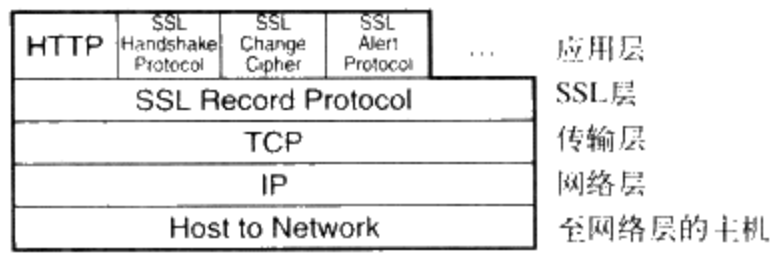

图18-3　SSL在协议堆栈中添加了一个附加层，与应用层一样控制自己的操作

SSL除了为HTTP提供安全传输环境以外，还能为其他协议提供安全传输环境。因为SSL层实质上是透明的，所以也可以使用其他协议。SSL层为下面的传输层提供接口，同样也为上面的应用层提供接口。SSL可以透明地处理握手、加密和解密。

当一个Web浏览器通过HTTP连接到一个安全的Web服务器时，两者需要通过一个握手协议来达成一些共识，例如，身份验证和加密。

握手序列包括下列步骤：

1）浏览器连接到一个启用了SSL的服务器并要求服务器验证自己。

2）服务器发送数字证书。

3）服务器有时可能（这种情况很少出现）要求浏览器证明其自身。

4）浏览器列出一组加密算法和它支持的哈希函数。服务器选择它所支持的功能最强大的加密算法。

5）浏览器和服务器生成会话密钥：

a) 浏览器从数字证书上获得服务器的公有密钥，并使用该公有密钥加密一个随机生成的数字。

b) 服务器对以简单文本形式发送的随机数据做出响应（除非浏览器已经为服务器的验证请求提供了数字证书，在这种情况下，服务器将使用浏览器的公有密钥）。

c) 用于会话的加密密钥通过使用哈希函数从随机数中产生。

生成高质量的随机数、解密数字证书、生成密钥，以及使用公有密钥加密系统都需要花费时间，因此这种握手过程也需要花费时间。幸运的是，结果将被缓存起来。如果同样的浏览器和服务器需要多次交换安全信息，握手过程发生一次，处理时间也只有一次。

当数据通过SSL连接发送时，将遵循下列步骤：

1）将数据分成易管理的数据包。

2）压缩每个数据包（可选的）。

3）每个数据包有一个信息验证码（MAC），信息验证码是通过哈希算法计算出来的。

4）将MAC和压缩的数据包组合到一起并加密它们。

5）加密后的数据与头信息组合在一起并被发送到网络中。

完整的过程如图18-4所示。

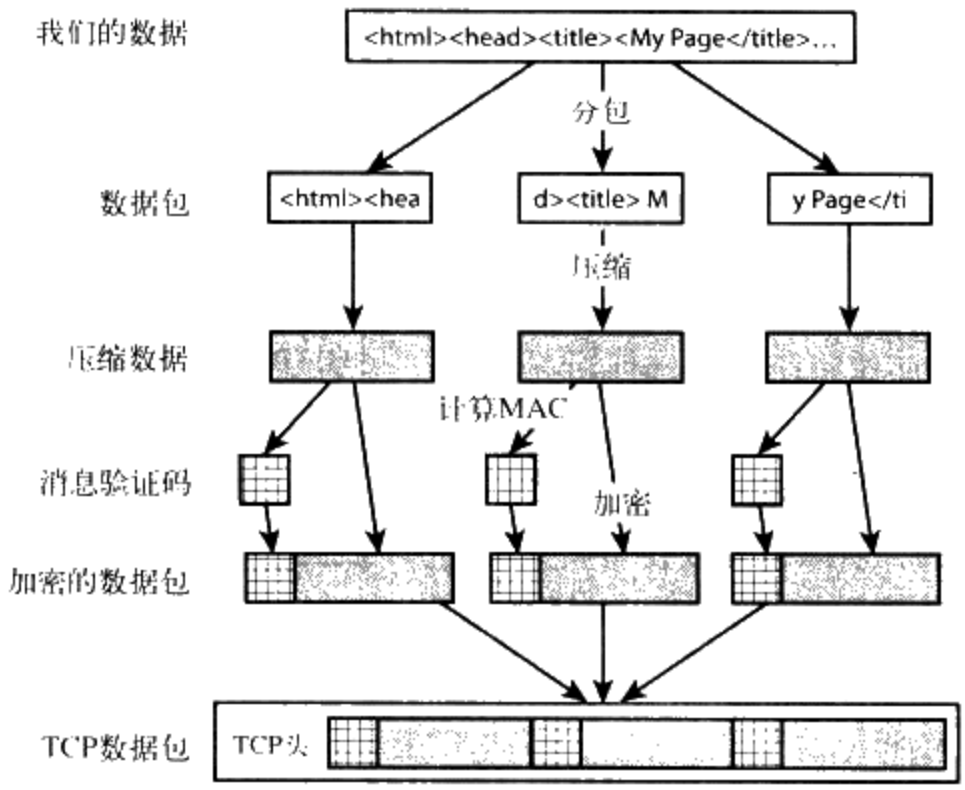

图18-4　发送数据之前，SSL将分解、压缩、哈希计算和加密数据

从上图中，我们会注意到，TCP的头信息是在数据加密之后才加入的。这就意味着在传输途中的信息仍然潜在地可能遭到修改，尽管刺探者不能辨别我们正在交换什么信息，但是他们能够看到谁正在交换信息。

SSL在加密之前包含压缩信息的原因是：尽管在网络传输中可以（通常是）压缩大多数网络通信，但是不能很好地压缩已加密数据。压缩模式依赖于数据的重复量和模式。在数据经过加密已经变成非常随机的组合之后，试图应用一种压缩算法通常是没有意义的。如果为提高网络安全性而设计的SSL带有极大地增加网络通信量的副作用，那么这是十分不合算的。

尽管SSL相对来说比较复杂，但是由于它的外部接口模拟现存的协议，因此在大多数情况下，用户和开发者可以受到保护。

目前，TLS（Transport Layer Security，传输层加密）还只是1.1版本，它是直接基于SSL 3.0标准的，但是还包含了一些可以克服SSL 3.0缺陷的改进，同时还提供了进一步的灵活性。人们计划将TLS设计为真正的开放标准，而不是由某个组织定义然后推广到其他组织。

18.3　屏蔽用户的输入

建立一个安全应用网站的原则之一是绝不可相信用户的任何输入，将用户数据输入到文件或者数据库之前，要一直屏蔽此数据或者通过系统执行的命令传输它。

我们已经在本书的几处地方讨论用于屏蔽用户输入的技术，在此，我们简要地列出这些技术，以作参考。

- 必须使用函数`addslashes()`在用户数据进入数据库之前过滤该数据。这个函数过滤掉可能引起数据库问题的字符。可以使用函数`stripslashes()`将数据返回到它的原始形式。
- 可以在php.ini配置文件中开启`magic_quotes_gpc`和`magic_quotes_runtime`的指

令。这些指令将自动地添加和过滤斜杠。magic_quotes_gpc指令用于格式化GET、POST和cookie变量，而magic_quote_runtime指令用于格式化进出数据库的数据。

- 当传递用户数据给system()或者exec()时，必须使用函数escapeshellcmd()。该函数可以避免任何怀有恶意的用户输入强迫系统运行某些特定命令的字符。
- 可以使用函数strip_tags()从一个字符串中去掉HTML和PHP标记。这样可以避免用户将恶意的脚本植入到用户的数据中（可能会将这些数据显示到浏览器上）。
- 可以使用函数htmlspecialchars()，该函数将字符转换成它们的HTML等价实体。例如，“<”被将转换成“<”，该函数可以将任何脚本标记转换成无害的字符。

18.4 提供安全存储

通常，3种不同类型的存储数据（HTML或PHP文件、与脚本相关的数据和MySQL数据）存储在同一磁盘的不同地方，图18-1中分别显示了这些数据。每一种存储类型的数据需要不同的预防措施，并且独立检查。

最危险数据类型就是可执行的内容。在网站上，可执行内容通常都是指脚本。需要注意的是，文件权限是否在网络安全层次里面设置正确。也就是，目录树是从Apache服务器上的htdocs目录，或者IIS服务器上的inetpub目录开始。为了看到脚本的输出，需要允许用户阅读脚本，但他们不能写入或编辑脚本。

同样的限制应用于Web层次结构的目录中。只有我们能够修改这些目录。其他用户（包括Web服务器运行的用户）不允许在这个目录中修改文件或者创建新文件，因为这个目录能够从Web服务器上下载。如果允许其他人在此目录中创建文件，他们就有可能创建一些恶意的脚本并通过Web服务器执行它。

如果脚本需要有写入文件的权限，就要在Web服务器的目录树结构以外创建一个目录。对于文件上传脚本，这是非常正确的。请注意，所创建的脚本和数据不要混在一起。当写入重要的数据时，可能首先要对它们进行加密，但是这种方式通常没有什么价值。我们这样考虑：如果在Web服务器上的文件名为creditcardnumbers.txt，一个黑客能够进入服务器并读取此文件，那它还能读什么呢？为了加密和解密数据，需要一个加密数据的程序，一个解密数据的程序，以及一个或者更多的密钥文件。如果解密高手能够读取数据，就没有什么能够阻止他读取密钥和其他文件。

只有当解密数据的密钥和软件没有存储在服务器上，而存储在另一台机器上时，Web服务器上的加密数据才可能有价值。安全处理重要数据的一个方法是在服务器上加密数据，然后将它传输到另一台机器，可以通过E-mail形式传输。

数据库的数据与数据文件类似。如果正确安装了MySQL，那么只有MySQL能够写数据文件。这意味着在MySQL中我们只要注意用户的访问权限就可以。我们已经讨论MySQL本身的权限系统，在特定的机器上它可以分配特定的权限给用户。

需要特别提出的是，我们经常需要在PHP脚本中编写MySQL密码。PHP脚本基本上是能够公开下载的。这并不是起初看起来那么严重的灾难。除非Web服务器系统崩溃，否则从外面来看PHP资源是不可见的。

如果Web服务器是利用PHP解析器根据扩展名.php来解释文件的，外面的人将不能够看到没有解析的源文件。然而，当使用其他扩展名时，应该小心。如果把.inc文件放到网站目录下，任何请求它们的人都可以收到没有解析的源文件。因此，需要将包含文件放网络之外，设定服务器不发送这种扩展名的文件，或者在这些文件中用.php作为扩展名。

如果与其他人共享一台Web服务器，MySQL密码对同一台机器上的其他用户可能是可见的，他们通过同样的Web服务器也能够运行脚本。根据系统的安装方式，这可能是不可避免的。服务器安装时设置为以单用户运行脚本，或者每个用户只运行自己的网站实例可以避免这个问题。如果你不是Web服务器的管理员（如果共享一个服务器就很可能出现这种情况），那么与你的管理员讨论这个问题，以及探讨安全选项是很有必要的。

18.5　存储信用卡号码

我们已经讨论了重要数据的安全存储，这里需要特别提出一种重要数据类型。Internet用户对信用卡号码是顾虑重重的。如果打算存储它们，需要非常小心。也需要问自己为什么打算这么做，是否真的必须这么做。

打算用信用卡号码做什么？如果要做一次交易处理和进行实时信用卡处理，最好从顾客那里接收卡号，然后直接发送此号码到交易处理网关而根本不用保存它。

如果有定期费用需要处理，例如，因为持续的订单而有权从一个信用卡每月索取一笔费用，上述方法可能不再是一个可选项。这种情况下，应该考虑将卡号存储到别处而不是Web服务器上。

如果打算存储顾客信用卡的详细信息，必须要有一个熟练并有些偏执的管理员，他有足够的时间去检查操作系统最新的安全信息资源，以及所使用的其他产品。

18.6　在PHP中使用加密技术

加密的一个简单但又实用的任务就是发送加密的电子邮件。多年来，为电子邮件进行加密的标准实际一直是PGP（Pretty Good Privacy）。Philip R.Zimmermann特别为电子邮件的保密编写了PGP。

我们可以获取PGP的免费版本，但是，需要注意的是，它并不是免费软件。这种免费版本只能供合法的非商业性机构和个人使用。

你可以从网站直接下载免费版本，或者从PGP公司购买一个商业许可证，请参阅如下网址获得详细信息：http: //www.pgp.com

关于PGP产品的历史以及可用版本，请参阅Philip Zimmerman在http://www.philzimmermann.com/EN/findpgp/findpgp.html上发表的“Where to Get PGP”文章。

作为PGP的替代，如今已经有一个开放源代码的类似产品可供使用。GPG（Gnu Privacy Guard）将是PGP的免费替代品。它不包含专利的算法，能够无限制地应用于商业活动中。

这两种产品以相似的方式执行同样的任务。如果要使用命令行工具，它可能没问题，但是二者具有不同的接口，例如，公共电子邮件程序的插件接口，一旦收到电子邮件，它将自动解密电子邮件。

GPG能够从以下站点获得：http: //www.gnupg.org

你可以同时使用这两种产品，使用GPG创建一个加密的消息，使用PGP（只要它是当前版本）对其进行解密。因为我们对于在Web服务器上创建消息很感兴趣，我们将在此给出一个GPG的例子。而如果使用PGP代替将不需要太多的变化。

与本书中例子的常规需求一样，需要获得能够让代码工作的GPG。系统中可能已经安装了GPG。但如果没有安装，也不用担心，因为安装过程非常简单，但设置GPG可能需要一点技巧。

18.6.1 安装GPG

要在Linux机器上添加GPG，需要从www.gnupg.org网站上下载适当的存档文件，并使用gunzip和tar命令从存档文档中提取文件。

要编译和安装程序，使用对大多数Linux来说都是相同的命令：

```
make
make install
```

configure（或者，./configure，这是因操作系统不同而各异）

如果不是root用户，需要使用带用--prefix选项的方式运行配置脚本，如下所示：

```
./configure --prefix=/path/to/your/directory
```

使用这个选项是因为一个非root用户将不能访问GPG的默认路径。

如果一切运行正常，GPG将被编译并复制到/usr/local/bin/gpg目录或者指定的路径。许多选项可以改变。详细信息请参阅GPG文档。

对于Windows服务器，这个过程是比较容易的。下载zip文件，解压缩它，将gpg.exe保存在PATH路径下（置于C:\windows\或者类似目录下是比较好的）。在C:\gnupg下创建一个目录。开启命令提示符模式并输入gpg命令。

还需要安装GPG或者PGP，并在系统上生成用以检查邮件的密钥对。

在Web服务器上，GPG和PGP的命令行版本之间几乎没有什么区别，因此我们也可以免费使用。在读取邮件的机器上使用GPG，最好购买PGP的商业版本，以便向邮件读者在邮件阅读器中提供一个更好的图形用户界面插件。

如果还没有这样的插件，请在邮件客户端机器上生成一个密钥对。回忆一下，密钥对包含一个公有密钥和一个私有密钥，其他人（和PHP脚本）在发送邮件之前使用公有密钥对邮件内容进行加密，我们可以使用私有密钥解密收到的信息或者签名要发送的邮件。

请注意，应该在邮件客户端机器上生成密钥而不是在Web服务器上生成它，这是很重要的。同样，私有密钥也不应该存储在Web服务器上。

如果使用GPG的命令行版本生成密钥，输入如下命令：

```
gpg --gen-key
```

系统将询问许多问题。大多数问题都有一个可以接受的默认答案。在不同的问题下，系统将询问真实名字和电子邮件地址以及说明信息，这些信息将用于命名密钥。例如，我的密钥就被命名为'Luke welling<luke@tangledweb.com.au>'。相信我们能够看懂此模式。如果给出了说明信息，那么说明信息将位于姓名和电子邮件地址之间。

要从新的密钥对输出公有密钥，可以使用如下所示命令：

```
gpg --export > filename
```

这将给出一个二进制文件，该文件适合将GPG或PGP密钥环导入到其他机器上。如果要将该密钥环以电子邮件的形式发送给人们，以便他们可以进入密钥环，那么可以建立一个ASCII版本来代替，如下所示：

```
gpg --export -a > filename
```

在已经获得公有密钥后，就能够将文件上传到Web服务器上自己的目录下。可以用FTP完成文件上传。

下面的命令是假定在UNIX环境下运行的。对于Windows系统来说，步骤也是一样的，但目录名和系统命令有所区别。首先，使用账户登录到Web服务器并更改文件的访问权根，以便其他用户能够阅读该文件。使用如下命令：

```
chmod 644 filename
```

我们需要创建一个密钥环，以便执行PHP脚本的用户能够使用GPG。具体哪些用户取决于服务器的设置。通常，该用户是“nobody”，但也可能是其他用户。

要想成为Web服务器用户，需要有访问服务器的root权限才能实现。在许多系统中，Web服务器都是以“nobody”用户身份运行的。如下所示的例子也是这种情况（我们能够在系统上将其改变为适当的用户）。如果系统是这种情形，可以使用如下所示命令：

```
su root
su nobody
```

为“nobody”创建一个目录用来存储密钥环和GPG的其他配置信息。这需要在nobody用户的根目录下进行。

每个用户的根目录在/etc/passwd文件中指定。在许多Linux系统中，nobody用户的默认根目录为“/”，而nobody用户无权对该目录执行写操作。在许多BSD系统中，nobody用户的根目录默认为/nonexistent（不存在），也就是说该目录不存在，用户也就不能对其执行写操作。在我们的系统上，nobody用户根目录被指定为/tmp。需要确定Web服务器用户是否有一个他们可以写入的根目录。

使用如下所示命令：

```
cd ~
mkdir .gnupg
```

nobody用户需要一个自己的签名密钥。要创建一个签名密钥，使用如下所示命令：

```
gpg --gen key
```

由于nobody用户可能需要接收一些个人电子邮件，因此可以为他们创建一个签名密钥。这个密钥的唯一目的是让我们信任先前获得的公有密钥。

为了导入先前导出的公有密钥，可以使用如下所示的命令：

```
gpg --import filename
```

要告诉GPG我们希望信任这个密钥，需要使用如下所示命令编辑密钥的属性：

```
gpg --edit-key 'Luke Welling <luke@tangledweb.com.au>'
```

在以上代码中，引号内的文本是密钥的名称，显而易见，当生成它时，密钥的名称不是'luke Welling<luke@tangledweb.com.au>'，而是由名称、注释和所提供的E-mail地址组成的。

这个程序的命令选项还包括help，它描述了所有可供使用的命令，例如，trust、sign和save。

输入trust命令选项将告诉GPG应该完全信任自己的密钥。输入sign命令选项表示用nobody的私有密钥签名公有密钥，最后，输入save命令选项表示退出这个程序并保存更改。

18.6.2 测试GPG

到这里，我们应该已经成功安装GPG并可以开始使用GPG了。创建一个包含一些普通文本的文件并以test.txt名字保存它，这样我们对其进行测试。

输入如下所示的命令（可以对该命令进行修改，用你的密钥名称来替换其中的密钥名称）：

```
gpg -a --recipient 'Luke Welling <luke@tangledweb.com.au>' --encrypt test.txt
```

将给出如下所示的警告：

```
gpg: Warning: using insecure memory!
```

并且生成名为test.txt.asc的文件。如果打开文件test.txt.asc，将看到如下所示的加密信息：

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.3 (GNU/Linux)
Comment: For info see http://www.gnupg.org

hQEOA0DU7hVGgdInEAQAhr4HdR7xpIBsK9CiFLQw85+klQdQ+p/FzqL8tICrQ+B3
0GJTBehPUDErwqUw/uQLTds0r1oPSrIAZ7c6GVkh0YEVBi2MskT81IIBvdo95OyH
K99UCvg/rLxdlkxe4Vp8QFET5E3FdII/ly8VP5gSTE7gAgm0SbFf3S9iPqwMyTkD
/2oJEvL6e3cP384s0i81rBbDbOUAAhCjjXt2DX/uX9q6P18QW56UICUOn4DPaW1G
/gnNZCkcVDgLcKfBjbkB/TCWWhpA7o7kX4CIcIh7K1MHY4KKdnCWQf271oE+8i9
cCRSCMsFIoI6MMNRCQHY6p9bfxL2uE39IRJrQbe6xoEe0nkB0uTYxiLGTG+FrNrE
tvBVMS0nsHu7HJey+oY4Z833pk5+MeVwYumJwlvHjdZxZmV6wz46GO2XGTl7b28V
wSBnWOoBHSZsPv<QXHTOq65EixP8y+YJvBN3z4pzdH0Xa+NpqbH7q3+xXmd30hDR
+u7t6MxTLDbgC+NR
=gfQu
-----END PGP MESSAGE-----
```

可以将该文件发送给最初生成此密钥的系统，运行如下命令：

```
gpg test.txt.asc
```

这样，就能够再次看到原始文本。原始文本将被写入到一个具有相同文件名称的文件——在这个例子中，文件名称为test.txt。

要将此文本输出到显示器上，可以使用-d标记并按如下方式指定一个输出文件：

```
gpg -d test.txt.asc
```

要在选择的文件中保存文本（非默认的文件名），可以使用-o标记并且指定输出文件，如下所示：

```
gpg -do test.out test.txt.asc
```

请注意，先要给出输出文件的名称。

如果已经安装了GPG，运行PHP脚本的用户就能够通过命令行使用它，基本上我们也是以这种方式运行。如果它出现问题，请询问系统管理员或者查看GPG文档。

程序清单18-1和程序清单18-2是通过PHP调用GPG来发送加密后的电子邮件。

程序清单18-1　private_mail.php——发送加密邮件的HTML表单

```
<html>
<body>
<h1>Send Me Private Mail</h1>

<?php
// you might need to change this line, if you do not use
// the default ports, 80 for normal traffic and 443 for SSL
if($_SERVER['SERVER_PORT']!=443) {
  echo '<p style="color: red">WARNING: you have not
        connected to this page using SSL. Your message could
        be read by others.</p> ;
}
?>

<form method='post' action="send_private_mail.php">
<p>Your email address:<br/>
<input type="text" name="from' size='40"/></p>

<p>Subject:<br/>
<input type="text' name="title" size="40"/></p>

<p>Your message:<br/>
<textarea name="body' cols="30" rows="10"></textarea></p>

<br/>
<input type='submit' name="submit" value="Send!"/>

</form>

</body>
</html>
```

程序清单18-2　send_private_mail.php——调用GPG和发送加密邮件的PHP脚本

```
<?php
  //create short variable names
  $from = $_POST['from'];
  $title = $_POST['title'];
```

```
$body = $_POST['body'];

$to_email = 'luke@localhost';

// Tell gpg where to find the key ring
// On this system, user nobody's home directory is /tmp/
putenv('GNUPGHOME=/tmp/.gnupg');

//create a unique file name
$infile = tempnam('', 'pgp');
$outfile = $infile.'.asc';

//write the user's text to the file
$fp = fopen($infile, 'w');
fwrite($fp, $body);
fclose($fp);

//set up our command
$command = '/usr/local/bin/gpg -a \\
            --recipient 'Luke Welling <luke@tangledweb.com.au>' \\
            --encrypt -o $outfile $infile";

// execute our gpg command
system($command, $result);

//delete the unencrypted temp file
unlink($infile);

if($result==0) {
  $fp = fopen($outfile, 'r');
  if((!$fp) || (filesize($outfile)==0)) {
    $result = -1;
  } else {
    //read the encrypted file
    $contents = fread ($fp, filesize ($outfile));

    //delete the encrypted temp file
    unlink($outfile);

    mail($to_email, $title, $contents, "From: ".$from."\n");
    echo '<h1>Message Sent</h1>
          <p>Your message was encrypted and sent.</p>
          <p>Thank you.</p>';
  }
}

if($result!=0) {
  echo '<h1>Error:</h1>
```

```
            <p>Your message could not be encrypted.</p>
            <p>It has not been sent.</p>
            <p>Sorry.</p> ;
    }
?>
```

为了使这些代码能够适用于你的情况，需要对它进行一些修改。电子邮件要发送到$to_email的地址中。

在程序清单18-2中，必须修改以下代码行：

```
putenv('GNUPGHOME=/tmp/.gnupg');
```

从而指定GPG密钥环的地址。在我们的系统中，Web服务器是以nobody用户的名义运行的，而且拥有一个根目录/tmp/。

我们使用函数tempnam()创建一个唯一的临时文件名。可以指定目录名和文件名的前缀。我们只需要很短的时间就可以创建和删除这些文件，因此如何命名这些文件并不重要。我们指定了一个前缀'pgp'，但是让PHP使用系统的临时目录。

如下语句：

```
$command = "/usr/local/bin/gpg -a \\
            --recipient 'Luke Welling <luke@tangledweb.com.au>' \\
            --encrypt -o $outfile $infile";
```

用来设置调用gpg的命令和参数。要将该命令适用于你的情况，必须对其进行一些修改。

在命令行上使用它时，需要告诉GPG哪一个密钥是用于加密信息的。

如下语句：

```
system($command, $result);
```

执行存储在$command中的命令，并将返回值存储于$result。我们可以忽略返回值，但是利用该值，我们可以使用一个if语句并且告诉用户某些语句出错。

当利用临时文件完成任务后，可以调用函数unlink()删除这些文件。这就意味着，未加密电子邮件在服务器上只存储了很短的时间。但是如果服务器在执行过程中失败了，文件保留在服务器上也是可能的。

当提到脚本的安全性时，对系统中的信息流进行考虑也是很重要的。GPG将加密电子邮件并允许接收者对其解密，但这些信息最初是如何从发送方发送过来的呢？如果提供一个Web界面来发送GPG加密邮件，那么其信息流与图18-5类似。

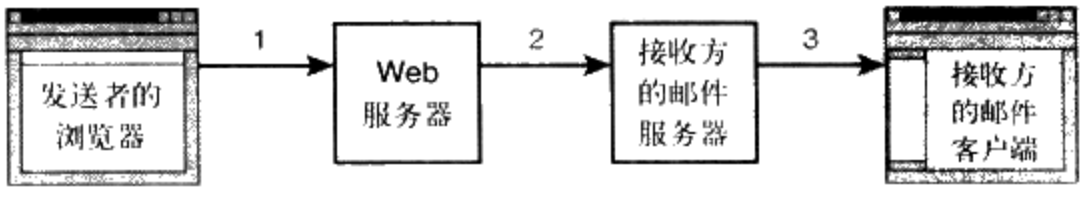

图18-5　在加密电子邮件的应用程序中，信息经过Internet发送3次

在本图中，第一个箭头代表消息从一台机器传输到另一台机器。消息每次发送后，它就在Internet上传输，并且中间可能经过许多网络和机器。

我们在此看到的脚本保存在上图中标记为服务器的机器上。在这个Web服务器中，消息将被收件人的公有密钥进行加密，并通过SMTP发送到收件人的邮件服务器。收件人连接邮件服务器（可能是利用POP或者IMAP），并通过邮件阅读程序下载此消息。然后可以用私有密钥解密消息。

如图18-5中所示的数据传输分别标记为1、2、3。在第2阶段和第3阶段中，传输的信息是一个加密的GPG消息，对任何没有私有密钥的人来说，该信息是没有价值的。在第1阶段中，传输的消息是发送者输入到表单中的文本。

如果信息非常重要，需要在上图的传输过程第2和第3阶段对消息进行加密，那么在第1阶段不加密是有些可笑的。因此，这个脚本属于使用SSL的服务器。

如果尝试不通过SSL连接这个脚本，该脚本将会给出警告信息。可以通过检查$_SERVER['SERVER_PORT']变量来确认。SSL连接的默认端口是443。任何其他的连接都将导致一个错误。

除了提供出错消息，我们还可以使用其他方式来处理这种情况。可以将用户重定向到一个经过SSL的URL，而且这个URL是相同的。我们还可以选择忽略它，这是因为如果表单是通过安全连接发送的，它通常就不是很重要的。通常，用户在表单中输入的具体内容安全地发送给我们才是重要的。我们可以将表单的动作指定为一个完整的URL。

目前，我们的form标记如下所示：

```
<form method "post" action="send_private_mail.php">
```

可以对其进行修改，使其通过SSL发送数据，即使用户没有使用SSL进行连接：

```
<form method="post" action="https://webserver/send_private_mail.php">
```

如果我们像以上代码那样，将完整的URL硬编码到form标记中，可以确保访问者的数据将通过SSL进行发送，但是当在另一台服务器上或在另一个目录下使用它时，必须对该URL进行修改。

尽管在这种情况以及许多其他情况下，通过SSL将一个空表单发送给用户并不是很重要的，但其实这样做却是一个非常不错的主意。看到浏览器的状态栏中的小挂锁符号，人们能够确信他们的信息将安全发送。他们无须查看HTML源代码中表单的动作属性是什么来确定数据是否安全。

18.7　进一步学习

SSL 3.0版本的规范说明可以从Netscape站点获得：http: //home.netscape.com/eng/ssl3/

如果希望了解网络和网络协议的工作原理，可以阅读由Andrew S. Tanenbaum编著的《Computer Networks》一书，这是一本介绍计算机网络和网络协议的经典图书。

18.8　下一章

到这里，我们将结束关于电子商务和安全问题的讨论。在下一篇中，我们将介绍一些更高级的PHP技术，包括：与Internet上其他机器之间的交互、动态图像的创建和会话控制的使用。