Web 应用程序安全设计探析

Security Design of Web Application

窦永富　崔为红　（山东省二轻工业经济技术情报所　250002）

摘要：Web 应用程序为结构设计人员、设计人员和开发人员提出了一系列复杂的安全问题。最安全、最有能力抵御攻击的 Web 应用程序是那些应用安全思想构建的应用程序。在设计初始阶段，应该使用可靠的体系结构和设计方法，同时要结合考虑程序部署以及企业的安全策略。本文提供了一系列安全的体系结构和设计方法，并按照常见的应用程序漏洞类别进行了组织。这些方法将对建设安全可靠的网站提供有益的帮助。

关键词：安全设计 Web 应用程序

1　Web 应用程序的体系结构和设计问题

Web 应用程序向设计人员和开发人员提出了许多挑战。HTTP 是无国界的，这意味着跟踪每位用户的会话状态将成为应用程序的责任。作为先导者，应用程序必须能够通过某种形式的身份验证来识别用户。由于所有后续授权决策都要基于用户的标识，因此，身份验证过程必须是安全的，同样必须很好地保护用于跟踪已验证用户的会话处理机制。设计安全的身份验证和会话管理机制仅仅是 Web 应用程序设计人员和开发人员所面临的众多问题中的两个方面。由于输入和输出数据要在公共网络上进行传输，因此还会存在其他挑战。

本文中的设计方法是根据应用程序漏洞类别进行组织的。实际经验表明，如果这些领域的设计存在薄弱环节，将会导致安全漏洞。下表列出了漏洞的类别，每个类别都突出显示了由于设计不当可能会导致的潜在问题。

由于设计不当导致的 Web 应用程序漏洞及潜在问题（如表 1）。

2　部署考虑

在应用程序设计阶段，应考虑公司安全策略和程序，以及部署应用程序的基础结构。通常，目标环境是固定不变的，应用程序的设计必须要反映这些限制条件。有时需要折衷考虑设计方案，例如，由于存在协议和端口限制，或是特定部署拓扑结构的要求。要在设计初期确定存在哪些限制条件，以避免日后在开发过程中出现意外。

2.1　安全策略和程序

表 1

漏洞类别	由于设计不当而引起的潜在问题
输入验证	嵌入到查询字符串、表单字段、cookie 和 HTTP 头中的恶意字符串的攻击。这些攻击包括命令执行、跨站点脚本（XSS）、SQL 注入和缓冲区溢出攻击。
身份验证	标识欺骗、密码破解、特权提升和未经授权的访问。
授权	访问保密数据或受限数据、篡改数据以及执行未经授权的操作。
配置管理	对管理界面进行未经授权的访问、具有更新配置数据的能力以及对用户帐户和帐户配置文件进行未经授权的访问。
敏感数据	泄露保密信息以及篡改数据。
会话管理	捕捉会话标识符，从而导致会话劫持及标识欺骗。
加密	访问保密数据或帐户凭据，或二者均能访问。
审核和记录	不能发现入侵迹象、不能验证用户操作，以及在诊断问题时出现困难。

安全策略确定允许应用程序及其用户执行哪些操作。更重要的是，安全策略定义了一些限制，以确定不允许应用程序及其用户执行哪些操作。设计应用程序时，应在公司安全策略定义的框架内进行标识和工作，以确保您没有违反防止部署应用程序的策略。

2.2　网络基础结构组件

确保您了解目标环境提供的网络结构，并了解网络的基本安全要求，如筛选规则、端口限制、支持的协议等等。

确定防火墙和防火墙策略可能会如何影响应用程序的设计和部署。在面向 Internet 的应用程序和内部

网络之间可能存在防火墙将其隔开。也许还存在用于保护数据库的其他防火墙。这些防火墙影响了可用的通信端口,因此会影响 Web 服务器到远程应用程序和数据库服务器的身份验证选项。

在设计阶段,需要考虑允许哪些协议、端口和服务从外围网络中的 Web 服务器访问内部资源。还应确定应用程序设计所需的协议和端口,并分析打开新端口或使用新协议会带来哪些潜在威胁。

2.3 部署拓扑结构

应用程序的部署拓扑结构和是否具有远程应用层是设计阶段必须考虑的关键问题。如果具有远程应用层,需要考虑怎样保护服务器之间的网络以减少网络窃听威胁,以及怎样保护敏感数据的保密性和完整性。

此外,还要考虑标识符流,并确定在应用程序连接到远程服务器时将用于网络身份验证的帐户。一种常见方法是使用最小特权进程帐户,并在远程服务器上创建一个具有相同密码的帐户副本(镜像)。另一种方法是使用域进程帐户,此类帐户管理方便,但会带来更大的安全问题,因为很难限制该帐户在网络上的使用。未建立信任关系的介入防火墙和单独域使应用本地帐户成为唯一的选择。

3 安全设计方法

3.1 输入验证

输入验证是一个很复杂的问题,并且是应用程序开发人员需要解决的首要问题。然而,正确的输入验证是防御目前应用程序攻击的最有效方法之一。正确的输入验证是防止 XSS、SQL 注入、缓冲区溢出和其他输入攻击的有效对策。

输入验证非常复杂,因为对于应用程序之间甚至应用程序内部的输入,其有效构成没有一个统一的答案。同样,对于恶意的输入也没有一个统一的定义。以下做法可以增强 Web 应用程序的输入验证。

(1) 假定所有输入都是恶意的。开始输入验证时,首先假定所有输入都是恶意的,除非有证据表明它们并无恶意。无论输入是来自服务、共享文件、用户还是数据库,只要其来源不在可信任的范围之内,就应对输入进行验证。

(2) 集中方法。将输入验证策略作为应用程序设计的核心元素。考虑集中式验证方法,例如,通过使用共享库中的公共验证和筛选代码。这可确保验证规则应用的一致性。此外,还能减少开发的工作量,且有助于以后的维护工作。

(3) 不要依赖于客户端验证。应使用服务器端代码执行其自身的验证。使用客户端验证可以减少客户端到服务器端的往返次数,但不要依赖这种方法进行安全验证。

(4) 注意标准化问题。数据的标准形式是最标准、最简单的形式。标准化是指将数据转化为标准形式的过程。文件路径和 URL 尤其倾向于标准化问题,许多广为人知的漏洞利用都直接源自标准化缺陷。

(5) 限制、拒绝和净化输入。输入验证的首选方法是从开始就限制允许输入的内容。按照已知的有效类型、模式和范围验证数据要比通过查找已知有害字符的数据验证方法容易。设计应用程序时,应了解应用程序需要输入什么内容。与潜在的恶意输入相比,有效数据的范围通常是更为有限的集合。

3.2 身份验证

身份验证是确定调用方身份的过程。许多 Web 应用程序使用密码机制对用户进行身份验证,用户可以在 HTML 表单中输入用户名和密码。以下做法可以增强 Web 应用程序的身份验证:

(1) 区分公共区域和受限区域。站点的公共区域允许任何用户进行匿名访问。受限区域只能接受特定用户的访问,而且用户必须通过站点的身份验证。

将站点分割为公共访问区域和受限访问区域,可以在该站点的不同区域使用不同的身份验证和授权规则,从而限制对 SSL 的使用。使用 SSL 会导致性能下降,为了避免不必要的系统开销,在设计站点时,应该在要求验证访问的区域限制使用 SSL。

(2) 对最终用户帐户使用帐户锁定策略。当最终用户帐户几次登录尝试失败后,可以禁用该帐户或将事件写入日志。

(3) 支持密码有效期。密码不应固定不变,而应作为常规密码维护的一部分,通过设置密码有效期对密码进行更改。在应用程序设计阶段,应该考虑提供这种类型的功能。

(4) 能够禁用帐户。如果在系统受到威胁时使凭证失效或禁用帐户,则可以避免遭受进一步的攻击。

(5) 不要在用户存储中存储密码。如果必须验证

密码,则没有必要实际存储密码。相反,可以存储一个单向哈希值,然后使用用户所提供的密码重新计算哈希值。为减少对用户存储的词典攻击威胁,可以使用强密码,并将随机 salt 值与该密码结合使用。

(6) 要求使用强密码。不要使攻击者能轻松破解密码。有很多可用的密码编制指南,但通常的做法是要求输入至少 8 位字符,其中要包含大写字母、小写字母、数字和特殊字符。无论是使用平台实施密码验证还是开发自己的验证策略,此步骤在对付粗暴攻击时都是必需的。

(7) 不要在网络上以纯文本形式发送密码。以纯文本形式在网络上发送的密码容易被窃听。为了解决这一问题,应确保通信通道的安全,例如,使用 SSL 对数据流加密。

(8) 保护身份验证 Cookie。身份验证 cookie 被窃取意味着登录被窃取。可以通过加密和安全的通信通道来保护验证票证。另外,还应限制验证票证的有效期,以防止因重复攻击导致的欺骗威胁。

3.3 授权

授权确定已通过验证的标识可以执行哪些操作以及可以访问哪些资源。错误授权或弱授权会导致信息泄漏和数据篡改。深入防御是应用程序授权策略的关键安全原则。

(1) 使用多重看守。在服务器端,可以使用 IP 安全协议 (IPSec) 策略提供主机限制,以此来限制服务器间的通信。IIS 提供了 Web 权限、Internet 协议/域名系统 (IP/DNS) 限制。无论用户是什么身份,IIS 的 Web 权限适用于所有通过 HTTP 请求的资源。如果攻击者设法登录到服务器,IIS 的 Web 权限将不提供保护功能。因此,NTFS 权限允许您为每个用户指定访问控制列表。最后,ASP. NET 提供 URL 授权和文件授权以及主要权限需求。将这些看守方法结合使用,可以制定出有效的授权策略。

(2) 限制用户对系统级资源的访问。系统级资源包括文件、文件夹、注册表项、Active Directory 对象、数据库对象、事件日志,等等。使用 Windows 访问控制列表 (ACL) 限制哪些用户可以访问哪些资源,以及可以执行哪些类型的操作。要特别注意匿名 Interne 用户帐户;使用 ACL 锁定这些匿名帐户,以防止他们访问明确拒绝匿名用户访问的资源。

3.4 配置管理

大多数应用程序需要使用接口,通过接口,内容开发人员、操作员和管理员可以配置应用程序和管理事项。如果支持远程管理,如何确保管理界面的安全?如果管理界面存在安全漏洞,结果会很严重,因为攻击者常常利用管理员特权中止程序运行,并能直接访问整个站点。

(1) 确保管理界面的安全。配置管理功能只能由经过授权的操作员和管理员访问,这一点是非常重要的。关键一点是要在管理界面上实施强身份验证,如使用证书。

如果有可能,限制或避免使用远程管理,并要求管理员在本地登录。如果需要支持远程管理,应使用加密通道,如 SSL 或 VPN 技术,因为通过管理界面传递的数据是敏感数据。此外,还要考虑使用 IPSec 策略限制对内部网络计算机的远程管理,以进一步降低风险。

(2) 确保配置存储的安全。基于文本的配置文件、注册表和数据库是存储应用程序配置数据的常用方法。如有可能,应避免在应用程序的 Web 空间使用配置文件,以防止可能出现的服务器配置漏洞导致配置文件被下载。无论使用哪种方法,都应确保配置存储访问的安全,如使用 Windows ACL 或数据库权限。还应避免以纯文本形式存储机密,如数据库连接字符串或帐户凭据。通过加密确保这些项目的安全,然后限制对包含加密数据的注册表项、文件或表的访问权限。

(3) 单独分配管理特权。如果应用程序的配置管理功能所支持的功能性基于管理员角色而变化,则应考虑使用基于角色的授权策略分别为每个角色授权。

(4) 使用最少特权进程和服务帐户。应用程序配置的一个重要方面是用于运行 Web 服务器进程的进程帐户,以及用于访问下游资源和系统的服务帐户。应确保为这些帐户设置最少特权。如果攻击者设法控制一个进程,则该进程标识对文件系统和其他系统资源应该具有极有限的访问权限,以减少可能造成的危害。

3.5 敏感数据

处理诸如信用卡号、地址、病例档案等用户私人信息的应用程序应该采取专门的步骤,来确保这些数据的保密性,并确保其不被修改。另外,实现应用程序时所用的机密数据(如数据库连接字符串)必须是安全的。在永久性存储中存储敏感数据以及在网络上传递

敏感数据时，数据的安全性是一个需要解决的问题。

3.6 会话管理

Web 应用程序基于无界限的 HTTP 协议构建，因此，会话管理是应用程序级职责。对于应用程序总体安全来讲，会话安全是关键因素。

以下做法可以提高 Web 应用程序会话管理的安全性：

- 使用 SSL 保护会话身份验证 Cookie
- 对身份验证 cookie 的内容进行加密
- 限制会话寿命
- 避免未经授权访问会话状态

3.7 加密

基本加密方式提供：

保密性（机密性）：此服务保持数据的机密性。

认可（真实性）：此服务确保用户不能拒绝发送特定消息

防止篡改（完整性）：此服务防止数据被修改

身份验证：此服务确认消息发送方的身份

Web 应用程序通常使用加密方式来确保永久性存储中的数据或在网络上传输的数据的安全性。在使用加密方法时，下列做法可以提高 Web 应用程序的安全性：

（1）不要自创加密方法。成功开发出加密算法和例程是非常难的。因此，应使用平台提供的经过验证和测试的加密服务。这包括 .NET 框架和基础操作系统。不要开发自定义的实现方法，因为这通常会导致保护能力减弱。

（2）将未加密数据存储在算法附近。向算法传递纯文本时，除非需要使用，否则不要获取该数据，并要以尽可能少的修改存储该数据。

（3）使用正确的算法和密钥大小。是否为一项作业选择了正确的算法非常重要，另外，应确保所使用的密钥大小能提供足够的安全级别。密钥越大，安全性越高。以下列表概括了主要算法及其使用的密钥大小：

- 数据加密标准（DES）64 位密钥（8 个字节）
- TripleDES 128 位密钥或 192 位密钥（16 或 24 个字节）
- Rijndael 128 - 256 位密钥（16 - 32 个字节）
- RSA 384 - 16,384 位密钥（48 - 2,048 个字节）

对于大量数据加密，应使用 TripleDES 对称加密算法。要减慢并加强对大量数据的加密，应使用 Rijndael 算法。要对将暂时存储的数据加密，可以考虑使用较快但较弱的算法，如 DES。对于数字签名，应使用 RSA 或 DSA 算法。对于哈希，应使用 SHA1.0 算法。对于用户键入的哈希，应使用基于哈希的消息验证代码（HMAC）SHA1.0 算法。

（4）确保加密密钥的安全。加密密钥是输入加密及解密进程的秘密数字。为保证加密数据的安全，必须保护好密钥。一旦攻击者得到了解密密钥，加密数据就不再安全了。

使用 DPAPI 来回避密钥管理和定期回收密钥将有助于确保加密密钥的安全。

3.8 审核和记录

应该审核和记录跨应用层的活动。使用日志，可以检测到踪迹可疑的活动。这通常能较早地发现成熟攻击的迹象，日志还有助于解决抵赖问题，即用户拒绝承认其行为的问题。在证明个人错误行为的法律程序中，可能需要使用日志文件作为证据。通常情况下，如果审核的生成时间恰好是资源访问的时间，并且使用相同的资源访问例程，则审核是最具权威性的。

以下做法可以提高 Web 应用程序的安全性：

- 审核并记录跨应用层的访问
- 考虑标识流
- 记录关键事件
- 确保日志文件的安全
- 定期备份和分析日志文件

4 结束语

安全性应渗透在产品开发生命周期的所有阶段，还应成为应用程序设计的关键问题。应特别注意可靠的身份验证和授权策略的设计。还应记住的是，大多数应用程序级攻击都源于恶意形式的输入数据和薄弱的应用程序输入验证设计。本文中提供的安全策略可以帮助解决在设计和构建安全应用程序中遇到的各种挑战，从而建设安全可靠的信息服务网站。

参考文献

1 Microsoft TechNet Web 应用程序安全设计指南。