Web 应用程序安全手册

Web 应用程序安全指南

由于网络技术日趋成熟，黑客们也将注意力从以往对网络服务器的攻击逐步转移到了对 Web 应用的攻击上。根据 Gartner 的最新调查，信息安全攻击有 75%都是发生在 Web 应用而非网络层面上。同时，数据也显示，三分之二的 Web 站点都相当脆弱，易受攻击。然而现实确是，绝大多数企业将大量的投资花费在网络和服务器的安全上，没有从真正意义上保证 Web 应用本身的安全，给黑客以可乘之机。

本技术手册为你全面解析 Web 应用程序安全问题，提供保证 Web 应用程序安全的方法和技巧。

Web 应用程序安全问题概述

相信大家都或多或少的听过关于各种 Web 应用安全漏洞，诸如：跨站点脚本攻击(XSS)，SQL 注入，上传漏洞等等。在这里我并不否认各种命名与归类方式，也不评价其命名的合理性与否，我想告诉大家的是，形形色色的安全漏洞中，其实所蕴含安全问题本质往往只有几个。

本部为你解读 Web 应用程序安全性问题的本质。

❖解读 Web 应用程序安全性问题的本质

如何保证 Web 应用程序安全

Web 应用程序是当今多数企业应用的前沿阵地。Web 应用程序在一个复杂的混合性架构中可以发挥多种不同的功能。管理与这些复杂的 Web 应用程序相关的风险是公司的必然要求，而且运行这些 Web 应用程序的底层代码的安全性直接影响到公司应用程序可用数据的风险态势。

本部分将为你介绍保证 Web 应用程序安全的方法和技巧。

- ❖使用 Burp Proxy 对 Web 应用程序进行调试和测试
- ❖Web 应用程序构建后的一些必备安全措施
- ❖如何保证 Web 应用程序安全性
- ❖保护 Web 应用程序不受直接对象引用（DOR）
- ❖Web 应用安全保护技巧

小心你的 WEB 应用程序成为数据窃贼的帮凶

当心，你以为固若金汤的数据库可能已遭到了入侵。你需要重新思考一下自己公司的网站是否真得不会遭到 SQL 注入攻击。SQL 注入是最流行也是最危险的 Web 应用程序漏洞利用技术，它可以攻击存储着珍贵企业信息的后端数据库，且“简约高效”。

本部分将阐述攻击者如何通过这种方法来利用 Web 应用程序的漏洞以及如何抵御这种攻击。

- ❖小心你的 WEB 应用程序成为数据窃贼的帮凶（一）
- ❖小心你的 WEB 应用程序成为数据窃贼的帮凶（二）
- ❖小心你的 WEB 应用程序成为数据窃贼的帮凶（三）
- ❖小心你的 WEB 应用程序成为数据窃贼的帮凶（四）

解读 Web 应用程序安全性问题的本质

相信大家都或多或少的听过关于各种 Web 应用安全漏洞，诸如:跨 site 脚本攻击(XSS)，SQL 注入，上传漏洞.....形形色色.。在这里我并不否认各种命名与归类方式，也不评价其命名的合理性与否，我想告诉大家的是，形形色色的安全漏洞中，其实所蕴含安全问题本质往往只有几个。 我个人把 Web 应用程序安全性本质问题归结以下三个部分：

1、输入/输出验证(Input/output validation)

2、角色验证或认证(Role authentication)

3、所有权验证(Ownership authentication)

说到这，读者一定想知道我这三种分类与形形色色的安全性问题有什么关系？下面我逐个给您概略解答：

输入/输出验证

这里的输入与输出其实都是发生在用户界面（User Interface）这一个层面上的，比如：你某一站点上提交一份注册信息，往往会收到诸多提示：“用户名非法”，“姓名不能使用英文“.....其实这就是输入验证的一个实例。什么情况是输出呢？ 比如说你成功提交一份注册信息后，系统会返回一个确认页（Registerred Confirmation），往往在这个页面上会显示你注册时提交的部分或全部信息，那么在这里显示的信息就是我所说的输出实例之一，输入需要做什么验证？假如你在提交时，在 Address 那一栏输入：

<script>alert("iwebsecurity");</script>, 当你到达注册的确认页时，会有什么发生？如果确认页没有做输出验证处理，那很显然会在到达确认页的时候出现一个 Javascript 打出的提示框。其实这就是跨 site 脚本攻击的一个小小的实例。当然了，单纯的输入/输出验证涉及的面可能够写一小本书了，努力在后续文章中给大家详解。

角色验证或认证

我们就拿 CSDN 来说吧，用户有这些角色：其一可以说是游客，就是浏览者没有登录时的角色；其二是免费的注册用户；或许将来 CSDN 深入发展了，业务有所更新，还会出现收费的注册用户。以上只是用户角色，那在 CSDN 公司内部还会有管理员角色，还有可能管理员又可以根据板块分为各种不同的角色。大家看到了吧，你天天访问的 CSDN 一共可能有多少角色？

接下来的问题就是权限问题了，为什么会有角色？就是为了控制权限的。每种角色都有自己特定的与公共的权限，这些权限的逻辑关系是相当复杂的，如果一个 Web 应用在角色上没有一个详细的合理的设计，将会给开发人员带来无限痛苦和麻烦。那现在我要问几个问题：你能保证每种角色只能做其份内的事儿？你是如何去保证的呢？方法可靠吗？有没有漏洞？.....这就是我要说的角色验证或认证。为什么我会说验证或认证呢？你可以这么理解，角色性存在于两个阶段，其一进入阶段，比如你登录的那一瞬间，你进入了一个特定的角色；另一个阶段就是维持阶段，你如何确保你登录后总是以登录时的身份在操作呢？那前者可以说是：认证，后者就是验证了。（有点罗嗦不？）

给一个角色认证 / 验证方面的虚拟案例，比如：一个在线电影服务提供商，会免费给您开一个试用角色，如果这试用角色验证不当，可能会导致用户权限提升而成为一个合法的收费用户，而这个收费用户你往往却收不到他的任何费用。

所有权验证

这个问题的存在也是基于角色的，只不过它所关心的是同级别的角色之间的权限问题。就拿 CSDN 来说吧，我是 CSDN 的一个免费用户，你也是。现在的问题是：我可以替你操作吗，我可以替你发表文章吗？我能修改你的个性设置吗？如果不能，CSDN 是如何实现的？虽然你和我都是普通用户，但是你有你的隐私我也有我的隐私，如何保证严格的所有权验证就显得尤为关键了。比较简单吧，这就是我所说的所有权验证。

我可以很自信的告诉你，只要是 Web 应用安全性问题，它逃不出在这三大部分，可能你还无法把形形色色的 Web 应用安全性问题与这三个部分对应并合理的解释清楚，但是确实只有这么简单的几个部分。如果您有疑问，可以以评论的方式提问。我可能会回复，也可能会以另一篇文章的形式出现，以供大家参考。

（作者：iwebsecurity　来源：TechTarget 中国）

使用 Burp Proxy 对 Web 应用程序进行调试和测试

问：什么是“Burp Proxy”？它是使用在大型企业环境中呢，还是仅仅为中小型企业以及消费者而设计的呢？

Burp Proxy 是免费版和专业版 Burp Suite 的核心部分，它是一个用于调试以及对网络应用程序进行安全检测的集成平台。从性能和安全的角度来看，它无疑可以在企业层面上发挥一定的作用，同时对于那些想要看清网络应用程序如何工作的人来讲，它也是一个有用的工具。特别是渗透测试者会发现它很有帮助，因为它是由一个渗透测试者开发出来的，所以它的许多功能适用于这类工作。

Burp Proxy 是一个交互式的 HTTP 和 HTTPS 协议服务器，它充当着浏览器和网络服务器的中间人角色。这意味着它可以拦截、查看和修改在这两者之间传递的流量。这种修改浏览器请求的能力，使得测试者可以发现应用程序如何工作并处理意外或者恶意的请求，比如 SQL 注入、cookie 破坏、会话劫持、目录遍历以及缓冲区溢出等。

它具有许多功能。例如，当图片和视频正在传输时，你可以处理它们的二进制数据。虽然输出包括了请求和应答的自动着色语法，但通过使用基于域、IP 地址、cookies、正文内容以及 HTML 页标题等各种参数的过滤器，使用者可以修改网络请求和响应，以此来控制哪些请求和应答需要被拦截下来进行人工测试。所有请求和做出的修改都保存在历史记录里，还可以保存到一个文件中，用于进一步的分析或者提供审核线索。

Burp Proxy 也与 Burp Suite 中的其它工具紧密集成，所以任何请求或者应答都可以被发送给其它工具用于进一步的处理。Burp Suite 的两个版本都包含一个用于映射网站的 spider 工具，它通过记录所有通过 Burp Proxy 发出的请求来建立一个详细的站点地图。由此产生的站点地图可以用于选择某些项目，并把它们发送给其它的 Burp 工具，用于分析或者把它作为一次攻击测试的一部分。这些工具可以在一起协同工作，这加快了人工或者自动测试的速度。

Burp 在 Linux 和 Windows 上均可运行，而且你还可以使用 IBurpExtender 接口来开发你自己的插件，从而拓展它的功能。专业版的价格是每个用户每年 275 美元，同时它包括一些额外的工具，比如一个应用程序漏洞扫描器，以及一个用于执行定制攻击来发现和利用罕见漏洞的入侵者工具。如果你有兴趣尝试使用 Burp 的话，可以在 PortSwigger 网站上下载免费的版本使用一下。 *(作者：Michael Cobb 译者：Sean 来源：TechTarget 中国)*

Web 应用程序构建后的一些必备安全措施

问：我们刚完成一个 WEB 应用程序的构建，我想知道你推荐使用哪些安全设备来保护它的正常运行。

答：在对你想要尝试保护的数据和设备充分了解前，我是不可能对你所需要的外围网络和应用程序数据安全设备做出详细具体的建议的。你可以参照一些基本步骤来为任何一种 WEB 应用程序所必需的安全特性准备一个简短的采购清单。

首先，将 WEB 应用程序所要使用到的数据进行分类是非常重要的，它将会存储在哪？它是如何进行存取和处理的？下一步，识别及评估这些数据以及处理这些数据的系统及应用程序的风险，这个过程被称做威胁建模，应该在应用程序设计过程中实现。通过从一个攻击者的角度来分析一个 WEB 应用程序，你将会对如下问题有更好的理解：

1. 它是如何受到攻击的？

2. 它为什么会受攻击？

3. 如何最大程度上减轻任何可识别的风险？

这个过程也可以帮助完善文档材料来识别和证明这个 WEB 应用程序的安全需求。

这些 WEB 应用程序的安全需求需要和整个组织全局性的安全策略相统一，这个全局性的安全策略定义了如何合法地保护这些数据的目标，基于此策略来决定如何最好地保护 WEB 应用程序，以防止受到任何可识别的威胁和降低敏感信息的风险。有一点我是非常肯定的，那就是如果重写部分应用程序的代码，逻辑及功能可以去除部分的漏洞，这样的努力应该通过附加的安全设备来进行补充，但是你的政策和策略必须明确一点，那就是你需要这些设备用来保护你的哪些数据以及用来防范哪些威胁。

当完成一个 WEB 应用程序后，你在查看用于减轻威胁的设备时，记得复查下它们用来保护防范哪种类型的威胁。有一些设备同时提供多种威胁的防护，比如病毒，间谍软件和恶意软件，而其它设备可能会着重于一种特定威胁，比如对即时聊天通讯上进行安全保护。你需要多加关注各家厂商在一种或多种安全领域所用到的这些技术覆盖的深度和方法。对应用程序数据带来多种攻击的一个常见问题是，它们经常在合法的客户端请求和响应上进行反复尝试，SQL

注入是一个很经典的例子，因此传统的外围保护技术，比如包过滤防火墙已经不再有足够的保护能力。

性能和可扩展性是另两个重要的考量点，某些安全设备可能受限于每小时可扫描的事务量，而另一些设备则可能有网络限制或仅对很小范围内的一些应用程序协议提供保护，我认为，何时选择一个安全设备应该先回答如下的关键问题：

1. 基于公司层面的安全策略目标和需求而言，它需要达到哪些目的？

2. 它对于现有网络的适应性如何？就目前所所拥有的技术力量能否正确有效地使用它？

3. 它将会对现存设备及用户造成什么影响，会造成怎么样的损失，比如设备重新采购，配置，人员培训所带来的花费等。

4. 它会提供哪些额外的服务？

很明显地，用于防护和处理数据的所有设备需要正确地安装。安装过程需要遵循一个包括四个步骤的安全生命周期：即安全、监控、测试和改进。这是一个持续的过程，一旦按照此流程完成，它将会在一个固定的保护周期中，不停地在这四个步骤中循环。在任何一台设备连接上生产网络前，确保它已经被加固过，打过补丁，同时进行进一步的安全配置。配置期间请确保参考你们自己的安全策略，用来保证这个设备被正确地配置以完成相应的任务，并且符合公司的安全方针。既然花了时间在选择和安装网络防护设备上，那么实施一个渗透测试是非常重要的，它可以确保这些设备的确按计划提供了这些保护。通过模拟这样的攻击，你可以评估你的站点是否还存在潜在的漏洞。

记住，你必须将基于渗透测试的结果所作的任何变更记录下来供以后参考，而且要确保配置没有被错误地改动，同时也必须做好对所有网络安全设备的物理访问控制和逻辑访问控制。如果仅限于依靠一些使用计量表类的安全设备，想达到保障应用程序安全是不可能的。你必须在各层次都建立防护措施：物理，网络，应用程序。通过使用这种威胁建模的流程可以保证在WEB 应用程序层面也是安全的，在增加了它们的强壮性外，也减少了对外围安全设备的依赖。

（作者：Michael Cobb　译者：陈运栋　来源：TechTarget 中国）

如何保证 Web 应用程序安全性

Web 应用程序是当今多数企业应用的前沿阵地。Web 应用程序在一个复杂的混合性架构中可以发挥多种不同的功能。其涉及范围极广，从在最新的云技术上运行的面向服务的方案，到较老的多层 Web 应用程序，再到准许客户访问大型主机上老应用程序的 Web 入口，都有其应用。

管理与这些复杂的 Web 应用程序相关的风险是公司的必然要求，而且运行这些 Web 应用程序的底层代码的安全性直接影响到公司应用程序可用数据的风险态势。不幸的是，开发可重复的高效 Web 应用程序的安全实践并非一项简单任务。许多单位试图运用后生产(post-production)解决方案来提供安全控制，如 Web 应用防火墙和入侵防御系统等。

但是一直等到生命周期的生产阶段才部署安全机制有点儿太迟，其效用也太小。设计或架构问题在生命周期的早期阶段更容易解决，如果等到应用程序投入生产之后再“亡羊补牢”，其所花费的成本将极其高昂。Web 应用程序的安全漏洞可导致数据泄露、违反策略，而且，在部署后再打补丁或进行全面的代码修复都会极大地增加总成本。

为保证效益和效率，Web 应用程序的安全必须从需求的定义阶段开始，直至最后的接收阶段。这种方法要求全体设计人员在整个过程中能够作为一个团队通力合作。在实施和测试等阶段，使用遵循策略的自动化工具能够支持可重复的测试，并且随着测试过程的标准化可以使开发周期更快。

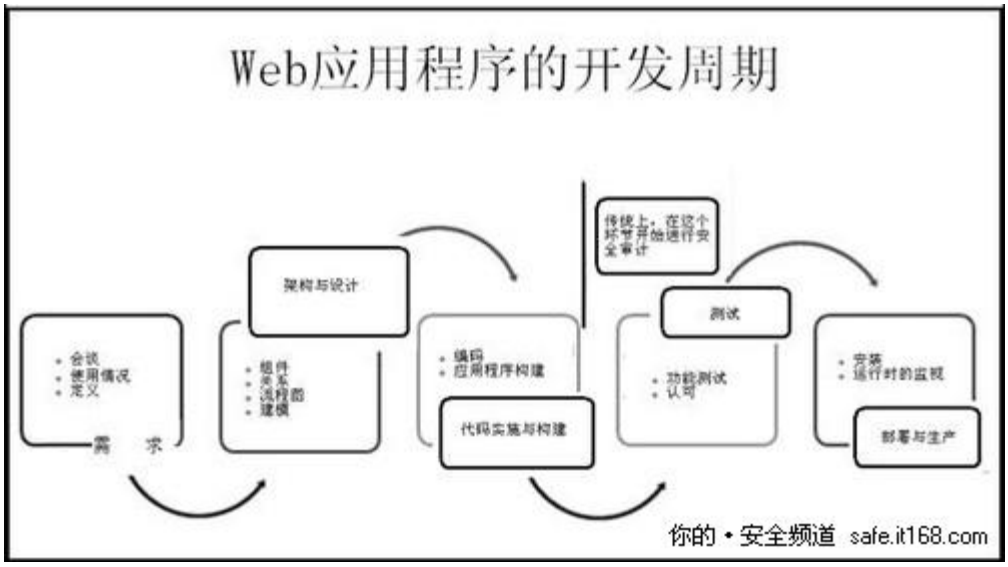
在开发过程的最初阶段就着手构建安全性未必很复杂。在整个开发周期实施安全检查和平衡，可以实现更快的发布周期，并可以极大地减少 Web 应用程序的漏洞。

在开发周期的后期实施安全测试的高昂成本

将安全检查点加入到开发过程中，确实可以减少总体交付时间。这听起来有点儿违反直觉，但是，在 Web 应用程序部署进入生产阶段之后，与纠正设计错误和代码错误相关的成本是极其高昂的。

例如，在许多开发环境中，安全和审核专家往往出现在开发周期结束之时。此时，应用程序已经完成，任何延误都被看作是多余的瓶颈。企业方面要求软件产品快速推出，这种巨大的压力意味着可能会忽略安全控制，导致 Web 应用程序没有经过恰当的安全审查。在这种时间极

端敏感的环境，即使扫描工具报告了大量的漏洞但却没有经过验证，也没有确定其优先次序，这种扫描也是弊大于利。



在开发过程的后期进行安全审计，而不是在整个开发周期中协力完成，会导致发布时间延迟，特别是在发现了某些严重的错误时尤其如此。在开发周期的后期修复设计和编码错误的成本远远高于早期发现错误所花费的成本。根据美国国家科学基金会的一项研究估计，如果在需求和设计阶段发现和纠正了严重的软件问题，其成本要比将软件投入生产之后再发现问题所花费的成本大约低 100 倍。

在 Web 应用程序中构建安全性时，在多数情况下，其目标并不是构建固若金汤的 Web 应用程序，或者清除每一处可能的漏洞。相反，目标应当是将所要求的特性与经认可的关于 Web 应用程序的风险预测匹配起来。在整个 Web 应用程序的开发周期中，其目标应当是实现软件担保，即与特定 Web 应用程序相适应的功能和敏感水平，能够有理由保证软件将会持续地实现其所要求的特性，即使软件遭受攻击也能够如此。

整合方法的好处

当来自不同小组的开发人员作为一个团队协同工作时，就会造就 Web 应用程序开发过程的高效率。虽然安全专业人员常常慨叹商务主管完全不理解软件风险，但是安全专业人员熟悉商业风险也很重要。通过适当的软件担保水平来构建 Web 应用程序，要求在商业需要、可用性和安全性之间进行风险管理权衡。为了达到适当的平衡，必须收集所有开发人员的需求。

从软件开发的开始阶段，需求定义和应用程序的设计就应当考虑安全需求以及功能需求和商业需要。在编写代码之前，这种信息就应当传达给设计师和软件开发人员。这种方法可以防止多数甚至是全部安全设计漏洞和架构漏洞。

然而，关注安全的软件设计并不是排除与 Web 应用程序相关的所有漏洞。开发人员自身必须接受关于安全编码技术的培训，以确保在应用程序的设计期间，并不会带来安全漏洞。让开发人员洞察开发语言和运行环境的安全方法可以支持更好的编码实践，并会使最终的 Web 应用程序出现的错误更少。将安全性纳入到设计过程中的另一个效率上的利益是，能够在需求和设计期间建立误用情形。 在测试和接收阶段这样做可以节省时间，并有助于消除瓶颈。

整合性合成测试的好处

软件测试的整合性的合成分析方法可以更大地提升效率。集成开发环境的特定插件在发现用户的编码错误时会向编码人员发出警告。静态分析，也称为“白盒”测试，在将不同的模块组装成最终的产品之前，开发人员和审计人员就对其使用此技术。静态分析在代码水平上提供了一种内部人员对应用程序的检查分析。静态分析对于发现语法错误和代码水平的缺陷是很有效的，但并不适于决定一个缺陷是否会导致可利用的漏洞。

动态分析和人工渗透测试对于验证应用程序是否容易受到攻击是有效的。它也被称为“黑盒测试”，主要展示的是外部人员对应用程序的检查分析，可以对投入生产的应用程序进行深入检查，查看攻击者是否容易利用其漏洞。然而，动态测试技术仅能用于软件开发的后期，只能在生成后阶段。动态测试的另一个局限性是它很难找出代码中导致漏洞的代码源。

这就是将静态测试和动态测试结合起来以“灰盒”或组合的方法进行混合测试的原因。通过将代码水平的内部检查和动态的外部检查结果结合起来，就可以充分利用两种技术的长处。使用静态和动态评估工具可以使管理人员和开发人员区分应用程序、模块、漏洞的优先次序，并首先处理影响最大的问题。组合分析方法的另外一个好处是动态测试确认的漏洞可以用静态工具追溯到特定的代码行或代码块。这便有利于测试和开发团队进行合作性交流，并使得安全和测试专家更容易向开发人员提供具体的可操作的纠错指导。

将安全性构建到软件的生命周期中：实用方法

构建安全性需要人员、过程以及技术、方法。虽然有大量的工具可有助于自动化地强化 Web 应用程序的安全，但是，如果没有恰当的过程和训练有素的人员来创建、测试 Web 应用程序，那么，任何工具都不会真正有效。

这个过程应当包括一个正式的软件开发周期及所公布的策略。此外，为所有的开发人员建立角色，并且指派检查和监管责任也是很重要的。安全和业务在软件开发周期的每一个阶段都应当得到表现，从而可以在每一个步骤处理风险管理。

在软件的整个开发周期，一个有益的永恒主题就是教育。教育对于开发人员是很重要的，它对于 Web 应用程序开发所涉及到的全体人员都很有益。因为安全意识既需要从上而下，也需要从下而上。不要低估教育管理人员认识到 Web 应用程序的漏洞如何影响企业的重要性。告诉一位管理人员 Web 应用程序易于遭受跨站请求伪造(CSRF)可能会令他感到茫然，但是如果向他展示软件错误如何会导致客户数据的泄露，就能够有助于使其意识到不安全的 Web 应用程序所造成的切实后果。应该准备特定的案例和度量标准用以阐明潜在的各种成本节约。例如，在开发人员检查其代码之前，就演示对开发人员的培训和 IDE 的静态分析插件投资可以阻止软件应用中的数据泄露根源。

审计人员和评估人员可以从学习常见的编码错误、后果评估以及与 Web 应用程序“生态系统”(包括后端的支撑系统、现有的安全控制以及属于 Web 应用程序环境的任何服务或应用程序)有关的依赖关系中获益。测试者及质量评价专家应当熟知误用情形，并且知道误用情形是如何区别于标准的应用的，还要知道如何解释安全测试结果，并能够按照需要对结果区分优先次序。

关注软件开发周期中的具体步骤，在这些步骤中存在着增加效率同时又可以贯彻安全性和风险管理的机会。

需求

Web 应用程序的设计者对于定义功能和业务需求非常熟悉，但是未必理解如何定义安全需求。此时，整个团队需要协同工作，决定哪些安全控制对最终的 Web 应用程序至关重要。

将安全性集成到需求阶段的步骤

1、根据公司策略、合规和规章要求，讨论并定义安全需求。

2、安全和审计团队应当评估业务需求和 Web 应用程序的功能，并且在测试和接受期间开始制定误用案例(误用情形)。

其好处有两方面，一是提前清除或减少安全或违规问题，二是减少部署时间。

架构和设计

由于已经定义了 Web 应用程序的架构和设计方案，下一步就需要评估安全问题。正是在这个阶段，成本高昂的、难以纠正的安全问题可以在其最易于解决的时间修复。为了防止损失惨重的错误，应当从性能和安全两个方面评估程序的架构。由于编制了详细的设计规范，因而可以向开发人员展示出应当包括哪些安全控制，以及应用程序的组件如何与完整的 Web 应用程序进行交互。

将安全性集成到架构和设计阶段的步骤

1、对于所建议的架构和部署环境执行风险评估，以决定设计是否会带来风险。

2、评估应用程序与原有系统进行交互时的安全意义和风险，以及不同的组件、层或系统之间的数据流的安全问题。

3、评述在实施或部署阶段需要解决的任何具体的暴露问题(即依赖于应用程序的部署方式和部署位置的漏洞)。

4、考虑依赖关系以及与混搭关系、SOA 及合伙服务的交互所引起的漏洞。将最终的设计交付安全和审计，以确定安全测试计划和误用情形。

其好处体现在五个方面：

1、可以精心协调风险评估分析过程和可重用的风险评估模型。

2、可以在早期阶段确定由架构环境或部署环境所带来的风险。

3、可重用的误用案例可以节省测试阶段的时间。

4、减少特定的设计漏洞。

5、如果有必要，可以调整或变更带来风险的架构限制，如果无法完全清除风险,也可以用补偿性控制来定义减轻风险的策略。

代码执行与编译

在开发人员开始编写代码时，他们对安全控制必须拥有一套完整的风险评估设计和清晰指南，或通过经认可的服务来使用这种安全控制。集成到 IDE 中的自动化静态代码工具可以在编写代码时向开发人员提供检查和指南。自动化的工具还可在编译期间用于对照策略模板检查代码是否违规，并可以深入查看代码水平的安全问题。

将安全性集成到代码执行和编译阶段的步骤

1、安装可与集成开发环境整合到一起的静态源码检查工具。

2、作为一种选择，开发人员在交付代码之前可以用独立的编码工具来执行自动化的代码检查。

3、安全和审核团队抽查代码模块，看其是否遵循合规要求，并在编译之前使用自动化的或手动的代码检查来检查其安全风险。

4、在编译过程中，执行自动化的静态代码扫描，查找安全问题和策略的遵循情况。

5、使用工具跟踪开发人员的编码错误，并对其带来的安全风险提供解释性反馈和原因说明。

这会带来如下的好处：

1、可将更清洁的或漏洞更少的代码提交给质量评价人员。

2、随着时间的推移开人人员可以提升安全编码能力。

3、可重用策略可以提升风险分析的正确性。

4、在测试阶段发现的编码错误或漏洞更少，并会使开发周期更短。

质量评价/测试

用于测试应用程序的具体安全工具范围很广，其中有可以评估完整应用程序的独立解决方案和服务，更有完全集成的套件,这种套件可以提供测试和对从教育到实施等多个阶段的支持。集成的方案可以为那些对可重复的 Web 应用程序的安全周期表现成熟的公司提供多个阶段的支持。集成套件可以在进程中的多个时点上实施，并可为持续的改善提供尺度和反馈。

那么，在质量评价和测试阶段，应采取哪些集成安全和改善安全的步骤呢?

1、专注于发现某些资源的最重要问题。

2、在一个包括现有的补救控制(如防火墙和 IPS)的应用架构中验证这些测试发现。

3、根据安全性和业务需要，区分所发现的漏洞的优先次序。

4、对于代码行或其所依赖的 API、服务、库等提出修复建议。

其好处也是显而易见的：1、应用程序开发人员之间可以更好地交流。2、似是而非的东西更少。3、修复周期更快。

部署/投产

Web 应用程序的安全性并不会终止于应用程序的部署阶段。一旦 Web 应用程序投产，还应当实施其它测试和监视，用以确保数据和服务受到保护。实际的 Web 应用程序的自动安全监视能够确保应用程序正在如所期望的那样运行，并且不会泄露信息从而造成风险。监视可由内部人员完成，也可外包给能够全天候监视应用程序的外部供应商。

在部署和投产阶段集成和改善安全性的步骤：

1、监视误用情况，其目的是为了确定测试中所谓的“不会被利用的漏洞”在投产后真得不会被利用。

2、监视数据泄露，其目的是为了查找被错误地使用、发送或存储的所有地方。

3、将部署前的风险评估与投产后的暴露范围进行比较，并向测试团队提供反馈。

4、实施 Web 应用防火墙、IPS 或其它的补救措施，其目的是为了减轻代码修复之前的暴露程度，或是为了符合新的安全规范。

其好处有如下几个方面：

1、改善能够成功实施的漏洞利用的知识库，从而改善在静态和动态测试期间的扫描效益。

2、发现并阻止应用程序的误用情况。

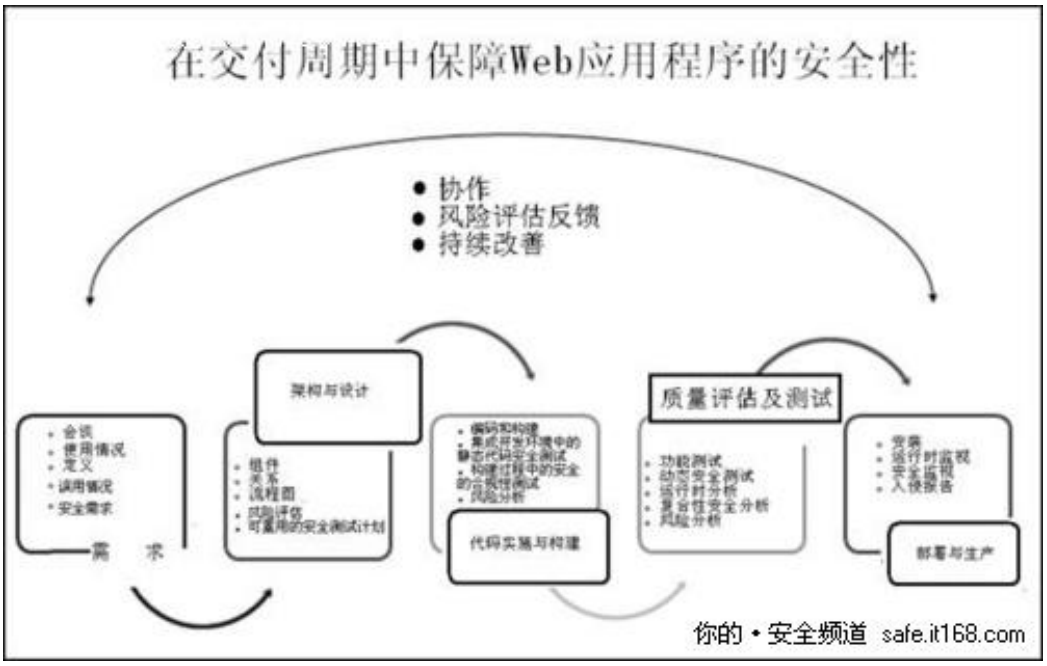
3、在动态测试和投产中的应用程序控制(如 Web 应用防火墙或 IPS)之间实现更好的集成。

4、满足再次编写代码之前的合规需要。

5、使用反馈机制实现持续的改善。

小结

保障 Web 应用程序的安全未必意味着延长开发周期。只要对所有开发人员进行良好的教育，并采取清晰且可重复的构建过程，企业就可以用一种高效而合作性的方式将安全和风险纳入到开发过程中。



将安全构建到 Web 应用程序的交付周期中确实需要一种合作性的方法，需要将人员、过程和技术集成到一起。虽然 Web 应用程序安全工具和套件有助于过程的改进，但它们并非万能药。为获得最大利益，应当考虑选择能够理解完整开发周期的 Web 应用程序安全工具厂商，还要有能在开发过程的多个阶段提供支持的工具。

（作者：建华　来源：TechTarget 中国）

保护 Web 应用程序不受直接对象引用（DOR）

指向特定文件、数据库记录或目录的 Web 应用程序容易遭受攻击。本文探讨对象引用如何影响企业网络安全，并讨论防御此攻击的方法。

不妨设想一个恶意黑客能够访问贵公司所有客户的账户细节，或者使用别人的信用卡在线购物，而这一切只需改变 URL 中的几个数字。这听起来似乎不太可能，但是如果你的 Web 应用程序容易遭受不安全的直接对象引用的危害，恶意黑客要达到这个目的简直易如反掌。

不安全的直接对象引用举例

这里的“对象”是指文件、目录、数据库记录等内部实施的对象，在应用程序将 URL（或表单参数）中的一个引用暴露给这些对象之一时，就会发生安全问题。这是因为黑客可以修改这些直接对象引用，例如，黑客可以在一个 URL 被提交之前进行参数修改，企图访问一个不同的、未获得授权的文件、目录，或数据库中的条目。如果不加强其它的授权检查，这种企图就会成功。

假设有一个 Web 应用程序最终会生成下面这个 URL：

http://www.yourinsecurewebapp.com/yourgetfile.cfm?filename=yoursometextfile.txt

这里有一个非常明显的对 yoursometextfile.txt 文件的直接对象引用。它对黑客的诱惑在于，看到如果将这个文件名换成另外一个文件名（如“yourpasswords.txt”或“youraccounts.txt”）会发生什么。

要取得这种成功，黑客必须正确地猜测出系统上另外一个文件名，但一个更合理的方法，是寻找系统上其它位置的特定内容，其使用的方法就是目录遍历攻击（目录遍历是 Http 的一个安全漏洞，它使得攻击者能够访问受限制的目录，并能够在 Web 服务器的根目录以外执行命令。）。从本质上讲，这意味着访问一个完全不同的目录，或者存在漏洞的应用程序的开发者所构建的任何方面。为访问 Apache Tomcat 文件名和口令，黑客可能将 URL 的最后一部分改成：

?filename=../../tomcat/conf/tomcat-users.xml

并非所有的直接对象引用都提供对文件的访问。还有另外一种可能激发黑客兴趣的URL，其结尾格式如下：

...account.cfm?customerid=4567

这会使黑客进一步问，“如果我将客户ID（customerid）换成4568会发生什么？”

与此类似，如果一个Web应用程序允许一个用户根据数据库的关键字引用从存储在数据库中的一个或多个信用卡中的一个，那么黑客修改此数据库的关键字时，会发生什么呢？

```
<select name="choosecreditcard">
    <option value="56">
        XXXXXXXXXXXX6902
    </option>
    <option value="88">
        XXXXXXXXXXXX5586
    </option>
</select>
```

在这里，用户可以从两个分别以6902和5586为结尾的卡中选择一个，该卡号由数据库的关键字引用，而应用程序可以访问此数据库文件。因此，黑客可以将56或88改为另一个数字，如78，用来引用属于另外一个用户的卡号。如果没有其它的认证检查来防止这种引用，攻击将获得成功。

避免不安全的直接对象引用

避免不安全的直接对象引用（DOR）漏洞的最佳方法是，完全不要暴露私密的对象引用，但如果非用不可，非常重要的一点是确保在向任何用户提供访问之前对其进行认证和审查。全球顶级的Web应用安全机构OWASP建议企业建立一种引用应用程序对象的标准方法，现简述如下：

1、尽可能避免将私密的对象引用暴露给用户，如重要的关键字或文件名。

2、运用一种“可接受的良好方法”，详细地验证任何私密的对象引用。决定准许用户访问哪些文件，并仅授与这些用户访问这些文件的权力。

3、对所有引用的对象都要进行验证。

OWASP 还提供了第三个要点的一个例子。在此，黑客可以将电子商务网站的购物车 ID 参数改为任何值：

```
int cartID = Integer.parseInt( request.getParameter( "cartID" ) );
String query = "SELECT * FROM table WHERE cartID=" + cartID;
```

要想防止受到这种攻击，就只能允许获得授权的记录可以显示：

```
int cartID = Integer.parseInt( request.getParameter( "cartID" ) );
User user = (User)request.getSession().getAttribute( "user" );
String query = "SELECT * FROM table WHERE
cartID=" + cartID + "AND userID=" + user.getID();
```

直接对象引用的另外一种选择是每个用户或会话都使用非直接的对象引用。

在前面那个关于信用卡的例子中，用户需要从两个卡中选择一个信用卡，这会暴露对信用卡数据库的直接引用。一个更好的方法是将这两个信用卡的记录存储到一个针对此用户的特定阵列中。关于信用卡的选择，其代码类似于下面的内容：

```
<select name=" choosecreditcard">
        <option value="1">
            XXXXXXXXXXXX6902
        </option>
        <option value="2">
            XXXXXXXXXXXX5586
       </option>
  </select>
```

在这种方法中，仅有对此用户阵列的一个直接引用，它仅包含此用户的数据。将选项的值改为大于 2 的任何值不会导致其他用户的信用卡细节被利用。然后，应用程序将把用户的特定的非直接对象引用（选项值为 1 或 2）映射回底层的数据库关键字（前面例子中的 56 和 88）。

对不安全的直接对象引用的测试

不幸的是，漏洞扫描器在发现不安全的直接对象引用漏洞方面并不是很高效，所以最佳的选择是：

1、仔细检查代码，确认是否有重要的参数易于遭到利用和操纵。

2、经常实施专业的渗透测试。

（作者：茫然　来源：TechTarget 中国）

Web 应用安全保护技巧

企业的员工通过安装一些未经 IT 管理部门审查的、控制或管理到的面向互联网的应用，不断地绕过公司的网络控制。其中最流行的应用软件有：即时通讯、基于 Web 的电子邮件、博客、播客、MP3 文件、P2P、VoIP 以及 PCAnywhere 这样的远程访问程序。

虽然这些应用程序对业务有好处，但同时也造成了技术和商业上的风险。比如，这些软件信息量丰富的特性(chatty nature)很可能导致数据泄露。此外，正如 Bank of America 的 Todd Inskeep 所说，“这些软件中的任何一个都有可能传播恶意软件的方法。”

金融巨头美洲银行的副总裁兼高级信息安全架构师 Inskeep 说，这些应用程序几乎都支持 80 和 443 端口上的连接，特别是像即时通讯软件这样的“端口不固定”软件，以及其他一些专有程序如 AOL Instant Messenger 可以频繁地通网络发送和接收信息。

Inskeep 建议，首先教育员工哪些东西允许下载，然后再安装多种软件管理工具,检测和移除桌面上不需要的应用。为了检查这些软件的使用情况，并确定怎样最有效的屏蔽它们，你可以考虑在隔离区（DMZ）里设置 Internet 连接来分析软件的协议和端口是如何改变的，这样就能避免反病毒或反间谍软件的检测。此外，还可以考虑扩大隔离区，从而同时利用内部和外部的防火墙：一个用于锁定端口，另一个用于分析数据包。

“如何作选择取决于你的业务需求” Inskeep 指出。美洲银行在全球范围内拥有 175,000 名员工，已经开发出了自己的企业即时通讯系统，这样既满足了沟通需要，同时也达到了监控的要求。为了提高生产效率并防止资产信息泄露，美洲银行不鼓励、也不禁止员工用这款软件联系朋友和家人。

想尝试一些新鲜的产品吗？Inskeep 建议安装 Skype 这款软件。这款免费的加密互联网电话软件会在内部一台独立的主机上主动扫描打开的连接。他说：“ Skype 可以像穿过瑞士奶酪中的洞一样穿过网络。”。

其他可选的措施包括：

- 锁定桌面，使用户不具有下载 Web 应用程序所需的管理员权限。
- 使用 URL 过滤，阻止那些可用于获得通讯或文件共享程序的网站。

- 分析 Web 应用程序所使用的协议，以确定哪些端口需要关闭。
- 启用常见的和（或）应用防火墙检测流量。
- 采取更强的策略，明确规定应该如何使用即时通讯、博客等服务。
- 对企业网络以外的设备，强制使用远程访问控制，如 VPN 等。
- 加强员工安全意识的培训，让他们知道哪些程序是可以用的，以及在什么场合下才可以用。

每个应用程序都需要进行风险分析。Inskeep 说：“大多数情况下，这样做的费用还是很低的。你的员工是免费下载软件的 。” 不过，增设安全措施将会导致费用增加。如果不能保护应用可能导致整个公司毁于一旦。

(作者：Anne Saita 译者：Sean 来源：TechTarget 中国)

小心你的 WEB 应用程序成为数据窃贼的帮凶（一）

当心，你以为固若金汤的数据库可能已遭到了入侵。你需要重新思考一下自己公司的网站是否真得不会遭到 SQL 注入攻击。SQL 注入是最流行也是最危险的 Web 应用程序漏洞利用技术，它可以攻击存储着珍贵企业信息的后端数据库，且“简约高效”。

本文将阐述攻击者如何通过这种方法来利用 Web 应用程序的漏洞。有时，即使攻击者也不了解自己正在利用的漏洞的性质。

何为 SQL 注入

就其最基本的意义来说，SQL 注入只不过是操纵一个已有的 SQL 查询，执行一个并非开发人员意图的动作。这种动作通常是通过 Web 应用程序的用户界面完成的。

但这种攻击是如何进行的？它为什么屡屡得逞？

Web 应用程序和数据库之间的正常交互

所有的 SQL 注入漏洞都是由某些未经验证的用户输入开始的。用户输入可以采取多种形式，它可以包括一个攻击者操纵的由服务器处理的任何东西，例如：用户代理、HTTP 报头、POST 参数、cookies、GET 参数，甚至网址标头等。是什么令未经验证的用户输入如此特殊呢？答案是：应用程序并没有对其进行充分的检查，从而不能确保所收到的输入就是所期望的类型和方式。例如，虽然你的应用程序的编制目的是为了接收可以包括字母、数字的字符串作为用户名，但此程序并没有验证输入，从而使得黑客可以插入 SQL 注入的数据库查询：

比如，一个典型的网站会要求你的用户名，并希望这个结果是“Zhangsan”：

Wangzhan.com/usertetail.asp?username=Zhangsan

在这个例子中，有可能会发生这样的情况：某个查询促使网站在后台与数据库交互，从而获取关于用户（如，Zhangsan）的信息：

```
SELECT uname,fname,lname,phone,street,city,state,zip FROM users WHERE user
=$var_username
```

当 Web 应用程序代码处理这个请求时，为了完成查询，来自用户名的值（Zhangsan）被传递给$var_username。服务器应当将 SQL 查询的结果变成标准格式，并显示此结果以便于用户查看 Zhangsan 的细节。

攻击者寻找突破口

首先，攻击者可能查看应用程序是否能够正确地处理错误条件。有许多方法可以检查 SQL 错误消息，每一个方法都依赖于数据库自身。最常见的例子是“ ‘”（ 撇号）。攻击者可能会尝试插入“ ‘”而不是一个合法的用户名：

Wangzhan.com/userdetail.asp?username= ‘

如果出现错误，攻击者就可以了解一些信息。例如，下面的错误就会使攻击者知道这是一个 MySQL 数据库，而且表明数据库将“ ‘”解释为查询的一部分，从而揭示出这可能是一个 SQL 注入点，值得进一步调查。

错误：您的 SQL 语法有一个错误，请检查您的 MySQL 服务器版本对应的手册，查看正确的语法…在第 4 行

在此例中，我们使用了一个“ ‘”，但任何“保留”字符，即在测试数据库错误时可以使用的为特别目的而保留的一个字符。保留字符对每种数据库类型来说都是独一无二的。

借助上面显示的 MySQL 错误消息，我们可以看出黑客是多么聪明，而且能够发现应用程序正在访问的数据库表的其它细节。请看：

Wangzhan.com/userdetail.asp?username=Zhangsan order by 1

如果我们没有收到错误，就可以知道用户名要么是 SQL WHERE 语句中的最后一个变量（允许我们从一个数据库表中重新获取数据，同时又排除其它的无关数据），或者是 WHERE 语句中的唯一变量。我们可以让数字每次增加 1，直至收到一个错误。例如，可能在到达“Zhangsan order by 9”，就可以看到：

错误：用户警告：“order clause”查询中有无法确认的列：SELECT

现在可以确认，直至提交了“9”，我们才收到了错误消息，所以可以断定表中有 8 列。这个信息很有用，但我们只是想获得尽量多的数据。假设没有提供输入验证，通过在用户名的位置使用一个通配符，我们实际上可以返回所有用户的细节：

Wangzhan.com/userdetail.asp?username=%

在该例中，我们将执行下面的查询，返回所有用户的细节：

SELECT uname,fname,lname,phone,street,city,state,zip FROM users WHERE user = %

如果攻击使用此伎俩，势必会造成数据损害，使大量的有价值的客户信息处于风险之中。其中可能包括应当被加密的用户口令，当然攻击者可以在日后再进行破解。遭到泄露的客户信息还有可能包括电子邮件地址，攻击者可以将其用于钓鱼攻击。

其实，我们可以不用插入简单的通配符，而是终止查询，并让查询做一些查询之外的事情：

Wangzhan.com/userdetail.asp?username=zhangsan;DROP users—

为便于比较，我们将 SQL Server 的数据库语句列示如下：

SELECT uname,fname,lname,phone,street,city,state,zip FROM users WHERE user = ‘zhangsan’;DROP users—

还要注意，此例允许你在同一行上提交多个查询（在此例中，即 SELECT 和 DROP 查询）。其方法就是用分号（；）分开并用两个破折号结束。因而，在完成最初的查询后，攻击者就可以发送并运行自己选择的一个完整查询。

(作者：茫然　来源：TechTarget 中国)

小心你的 WEB 应用程序成为数据窃贼的帮凶（二）

在上一篇文章中，我们介绍了攻击者如何通过 SQL 注入攻击来利用 Web 应用程序的漏洞。本文我们将继续介绍 SQL 注入攻击的两种方法，并举例自动 SQL 注入。

自动攻击

攻击者可以使用两种主要的攻击方法来利用 SQL 注入漏洞：自动攻击和手动攻击。从字面上看似乎很容易理解，但这两种攻击在机制上是非常不同的。自动攻击一般是为特定目的而编制的一种工具所导致的结果。例如，有的攻击使用大规模的注入攻击，其目的是为了让其攻击范围最大化并任意扩散其代码。这种大规模的攻击一般针对非常具体的应用程序架构，如运行 ASP 的 IIS 服务器等。而 Asprox 攻击，其目的是为了将一个 JavaScript 或 iframe 标记注入到网页中，从而传播病毒。

自动 SQL 注入举例

```
wangzhan.com/ssp.asp?username=zhangsan’;DECLARE @T VARCHAR (255),@C
VARCHAR(255) DECLARE TABLE_CURSOR CURSOR FOR SELECT A.NAME,B.NAME FROM
SYSOBJECTS A,SYSCOLUMNS B WHERE A.ID=B.ID AND A.XTYPE=‘U’ AND (B.XTYPE=99 OR
B.XTYPE=35 OR B.XTYPE=231 OR B.XTYPE=167) OPEN TABLE_CURSOR FETCH NEXT
FROM TABLE_CURSOR INTO @T,@C WHILE(@@FETCH_STATUS=0) BEGIN
EXEC(‘UPDATE [‘+@T+’] SET
[‘+@C+’]=RTRIM(CONVERT(VARCHAR(4000),[‘+@C+’]))+’<script code>‘)
FETCH NEXT FROM TABLE_CURSOR INTO @T,@C END CLOSE TABLE_CURSOR DEALLOCATE
TABLE_CURSOR;--
```

下面，我们简单地看一下这段代码是如何针对后端数据库实施攻击的。首先，攻击者声明了 Table (T)和 Column (C)这两个变量。

```
DECLARE @T VARCHAR(255),@C VARCHAR(255)
```

并声明了一个可以保存查询结果的表 cursor：

```
DECLARE TABLE_CURSOR CURSOR FOR
```

下面的 SELECT 语句通过“text”、“sysname”、“varchar”等列来检索所有的用户对象，并且将结果存储在前面创建的 CURSOR 中。

```
SELECT A.NAME,B.NAME FROM SYSOBJECTS A,SYSCOLUMNS B WHERE A.ID=B.ID AND
A.XTYPE= ‘U’  AND (B.XTYPE=99 OR B.XTYPE=35 OR B.XTYPE=231 OR B.XTYPE=167)
```

在下面的代码中，数据表 CURSOR 检索结果，并将其分配给表和列变量：

```
OPEN TABLE_CURSOR FETCH NEXT FROM TABLE_CURSOR INTO @T,@C
WHILE(@@FETCH_STATUS=0)
```

此时，攻击者已经检索了数据库中基于文本的这些列，其意图是为了修改这些列的内容。在这里，攻击者虽然没有篡改数据，但已经完成了所有必要的侦察。然后，执行更新语句，将 JavaScript 置于列变量中的每一列中。攻击完成后，包含 Web 内容（这些内容源自数据库的任何字段）的任何网页都会提交攻击者的恶意 JavaScript 代码。然后，该 JavaScript 用一个病毒感染 Web 用户的计算机：

```
WHILE(@@FETCH_STATUS=0) BEGIN EXEC( ‘UPDATE [ ‘+@T+’ ] SET
[ ‘+@C+’ ]=RTRIM(CONVERT(VARCHAR(4000),[ ‘+@C+’ ]))+’<script code> ‘) FETCH NEXT
FROM TABLE_CURSOR INTO @T,@C END
```

在注入的结尾，攻击者执行清理，覆盖所有的攻击痕迹：

```
CLOSE TABLE_CURSOR DEALLOCATE TABLE_CURSOR;--
```

这些攻击是完全自动化的；黑客攻击只需使用搜索引擎简单地搜索互联网，查找运行经典的 ASP 代码的 Web 服务器即可。别再自欺欺人的相信“我的网站很小，谁会愿意攻击它呢？”这种愚蠢的谎言。如果你正在通过互联网做商务，不管企业大小，都易于遭到攻击。

接管

在很多情况下，攻击者能够完全控制 SQL 服务器的底层操作系统；攻击者甚至可以接管 Web 应用程序，并最终接管 Web 服务器。

接管数据库服务器可以导致损害其它的应用程序，甚至损害 DMZ 中的其它服务器。现代的 Web 应用程序架构一般都有数据库集群，与其它系统共享数据存储，或者位于网络中不太安全的地方。如果攻击者考虑到这个方面，他就可以使用前面提到的方法绕过防火墙中一般的 IP 源过滤规则，从而攻击内部网络，甚至还可以使用 SQL 服务器来存储病毒、黄色图片或其它非法内容。

此外，通过首先接管数据库服务器，攻击者可以篡改 Web 应用程序的行为。通常，这种行为包括借助 Web 服务器的服务账户在本地服务器上运行命令。如果服务账户有了被提升的特权，攻击者就可以通过数据库服务器，将命令直接发送给 Web 服务器的操作系统。

有时，数据是根据计划例程从 DMZ 数据库服务器析取出来的。如果通过公司 Intranet（内联网）中的 Web 接口来查看数据，情况就更危险，因为多数内联网的 Web 应用程序在运行时，其信任等级更高。

不要小看盲目攻击

通常，控制框架会把 SQL 错误搞得不易分辨，如 Java 或.NET 框架。有时，这些错误是由错误处理代码自动处理的，或者是由底层的代码解释程序处理的。例如，攻击者可以使用撇号（‘）而不是合法的输入：

wangzhan.com/userdetail.asp?username=‘

其结果为：用户未找到。

这就告诉我们，Web 应用程序正在正确地检查用户输入，或者是程序框架正在阻止显示明显的错误消息。在这种情况下，执行 SQL 注入就变得困难了，但并不是不可能。这就引出了下一种攻击：盲目 SQL 注入。

盲目 SQL 注入攻击是如何工作的呢？它要查看数据库服务器是否真正地处理请求，或者查看该请求是否会触发来自 Web 服务器或 SQL 服务器的其它响应。

例如，我们已经看到了能够向我们提供张三（Zhangsan）用户细节的请求，能够看出该攻击是否成功。不过，“WAITFOR DELAY”命令要求 SQL 服务器在用查询结果响应之前，先暂停一分钟。

wangzhan.com/userdetail.asp?username=zhangsan;WAITFOR DELAY ‘0:1:0’—

如果网站能够正确地处理用户输入，它应当返回一个消息，说明找不到用户，或者发出其它的通知，说明“zhangsan; WAITFOR DELAY ‘0:1:0’--”不是一个合法用户。

必须指出，只有在将用户名变量的值被提交给 SQL 服务器时，并且 Web 应用程序能够正确地处理用户名变量的值时，才会出现这种情况。

传统的 SQL 注入将会返回错误，让用户知道有关查询失败的原因的细节。从本质上讲，盲目 SQL 注入依赖的是一种来自服务器的布尔逻辑响应（是/否，真/假等）：查询请求要么被处

理，要么遭遇失败。盲目 SQL 注入攻击一旦得以发现，它就会提供与典型的 SQL 注入一样的功能，但它更难以执行，为了获取信息也需要更多的时间。

让盲目攻击起作用

假设在示例页面中，我们已经发现了盲目注入，我们希望发现关于表和数据库设计的细节。请看下面的请求：

```
wangzhan.com/userdetail.asp?username=zhangsan AND
ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE
xtype=0x55),1,1)),0)>65—
```

如果我们分解一下，会看出这个查询的结果实际上是一个是或否的回答，“这个表名的第一个字符是一个比 65 大的 ASCII 值吗？”先看最里面的查询：

```
SELECT TOP 1 name from sysObjects WHERE xtype=0x55
```

这意味着：获取来自 sysObjects（所有的表）列表最上面的一个名字，其 xtype（对象类型）等于 ASCII 码 0x55（大写的字母“U”），转换成用户（user）表。所以，基本而言，我们获取了用户在数据库中所创建的第一个表。为简单起见，假设这个表就是“users”：

```
SUBSTRING(users),1,1
```

现在，我们创建了一个子串，它包含了表名 users 的首个字符。其值为“u”：

```
ISNULL(ASCII(u),0)>65
```

在这里我们查看 u 的 ASCII 码值的结果是否大于 65，确认表名以字母开头。如果答案是肯定的，查询就会被处理。否则，查询就不会被处理。

使用这种方法，我们可以对每个查询简单地增加数字，从而发现首个用户名的第一个字母。例如，假设我们在大于 117 时（>117）得到了一个否定的响应，我们知道第一个字母是字母“u”，因为“u”的 ASCII 值是 117。在发现这一点之后，我们可以简单地改变 SUBSTRING 参数来选择第二个字母，然后是第三个字母，等等。

最后，在没有错误代码时，我们借助于盲目注入就可以发现有漏洞的 Web 应用程序的数据库的完整的表结构。

（作者：茫然　来源：TechTarget 中国）

小心你的 WEB 应用程序成为数据窃贼的帮凶（三）

在（一），（二）前两部分文章中，我们介绍了攻击者如何通过 SQL 注入攻击来利用 Web 应用程序的漏洞以及 SQL 注入攻击的方法。本文我们将介绍抵御该攻击的措施。

防御措施：保证应用程序编码的安全

安全的编码技术可以清除造成 SQL 注入攻击的漏洞。下面说的是 Web 应用程序安全编码的三个基本方法。

方法一：输入验证

在 Web 应用程序这一层防止 SQL 注入最常见的方法是使用输入验证。无论是何种语言或平台，这种方法都很有用。其实质就是在验证用户输入的大小和类型前，不对其采取行动。如果你期望用户输入的是数字，就不要接受非数字的东西。

例如，下面的 URL：

wangzhan.com/userdetail.asp?userid=9899

很明显，在这个 GET 请求中，我们期望输入一个整数。一个简单的类型检查会告诉我们这是不是一个合法的字符，从而确保应用程序不会处理非数字值的输入。

```
if (is_numeric($userid)){}
```

此外，如果用户的 ID 总是四个字符的长度，我们就可以进一步采取措施，除了使用常规的表达式来执行整型检查，还可以强化边界检查。

```
if (preg_match( '\d{4}' , $string)) {}
```

在对待如何验证用户输入值的这个问题上，我们仅受到自己创造力的限制。这里的关键是验证由用户发送并由系统使用的每一个值。

执行类型检查的另外一种方法是通过 ASCII 字符。用户名一般是由字母和数字组成的，所以我们不希望看到类似“@”、“；”之类的字符。因而，我们可以解析用户名变量，看看是否存在着不属于 48-57,65-90，97-122 的任何 ASCII 字符。在这个范围之外的任何 ASCII 字符对于用户名变量来说，都是不合法的，应当拒绝接受。这就是所谓白名单的一个例子。

知道白名单和黑名单的区别非常重要。白名单仅接受已知为安全的值或字符，如字母和数字。而黑名单则会阻止或不接受已知为恶意的字符。

方法二：规避技术

到目前为止，类型检查和边界检查似乎很容易，但并非在检查所有的数据类型时都会这么简单。许多情况下，我们还会使用 VARCHAR 数据类型（VARCHAR 是一种比 CHAR 更加灵活的数据类型，同样用于表示字符数据，但是 VARCHAR 可以保存可变长度的字符串。）此类型有可能包含危险字符。这在备注字段和其它的长表单文本字段中很常见。

在这种情况下，我们可以利用一种称为规避的技术来确保变量的内容绝对不会被解析为 SQL 语句的一部分。

请看下面这个例子中的请求：

wangzhan.com/comment.asp?msg=I’m zhangsan.

在这个例子中，有一个撇号（’），这是一种常被认为是恶意字符的字符（请参考上一篇文章提到的撇号），我们并不希望排除它，因为此处它的用法是合法的，我们也不希望拒绝这个消息。

例如，在 PHP 中，我们可以使用 mysql_real_escape_string 函数：

mysql_real_escape_string($GET[‘msg’]);

这就不会导致安全问题，而是安全地解决了撇号问题，使其不能用于任何 MySQL 查询中：

msg=“Hello I\’m zhangsan”

同样地，在微软的.NET 架构中，设计者常用 REPLACE 来保证字符串的安全。在下面的例子中，REPLACE 函数将撇号（’）放在引号中，使其成为一种安全字符：

sql = replace(str, “’ “, “’ “)

方法三：参数化查询

防止 SQL 注入的第三种方法是一种称为参数化查询的技术。这种方法非常有效，因为它可以非常严密地控制 SQL 语句的组成结构。此方法在将对 SQL 语句的任何重要变更交给 SQL 服务器处理之前就拒绝其操作。

下面的 Java 例子中，我们简单地将参数添加到已经构建的静态查询中。首先，我们使用问号作为变量建立了真实的 SELECT 语句：

```
String query = “SELECT account_balance FROM user_data WHERE user_name = ? “;
```

下一步，我们调用 prepareStatement 函数：

```
PreparedStatement pstmt = connection.prepareStatement( query );
```

然后，在查询中我们将字符串“custname”指派给变量：

```
pstmt.setString( 1, custname);
```

最后，我们执行查询，并将结果存储在一个变量中：

```
ResultSet results = pstmt.executeQuery( );
```

通过使用这种方法，我们能够确保在附加的恶意查询或参数被发送给数据库之前，不会被添加到查询中。注意，应当像前面所讨论的那样来验证“custname”变量，因为它是不受信任的用户输入。

在下一篇文章中，我们将讨论另外一种策略：使用存储 SQL 过程。

(作者：茫然　来源：TechTarget 中国)

小心你的 WEB 应用程序已成为数据窃贼的帮凶（四）

在系列文章（三）中，我们介绍了抵御 SQL 注入攻击的安全编码技术。并详细介绍了 Web 应用程序安全编码的三个基本方法，包括：输入验证，规避技术，参数化查询。本文我们将继续介绍其他的方法。

另一种策略：使用存储 SQL 过程

你完全有可能无法控制特定 Web 应用程序的输入验证和控制。在许多情况下，SQL 的开发在完成后就不再改变了，而 Web 应用程序的代码却要经历多个版本。

你也许没有通过数据库来控制安全的能力，不过还是有办法的。使用存储过程可以给数据库管理员一些防止 SQL 注入的方法。例如，在微软的 SQL 中，创建存储过程是很简单的。在此，不妨使用我们前的的例子：

```
wangzhan.com/comment.asp?msg=ni hao ma
```

我们要创建一个过程，仅期望自己的消息内容是：

```
CREATE PROCEDURE sp_addComment
@msg nvarchar(30)
AS
INSERT INTO comments_table
(
msg
)
VALUES
(
@msg
)
GO
```

执行时，可以这样做：

```
EXEC sp_addComment @msg = $msg
```

使用这种方法，我们就可以确保不会执行“意外”的查询。如果$msg 变量中的值不满足过程的数据类型，就会出现一个错误，但不会产生 SQL 注入的条件。

增强最小化特权

最小特权的原则要求：任何一层的 Web 应用程序所使用的任何账户，都不应当拥有超过其为了执行特定职能而必需的更多特权。例如，许多 MySQL 的版本都用根特权来运行。这样做的风险很大，因为 MySQL 实例所运行的任何系统命令，都会以最高等级的许可来运行。其实，在几乎所有情况下，MySQL 实例为了完成其工作，仅需要几种特定的许可。

同样的原则还可应用于不同的层相互会话的方式。例如，应用程序有可能使用一个 SQL 用户账户来对数据库进行操作。在很多情况下，我们可以在表甚至在列的水平上将特定的许可指派给这个用户账户，防止其被恶意使用后所导致的损害。

如果不允许用户表被 Web 应用程序服务账户删除，就应当从用户那里移除这些许可。如果不允许服务账户修改特定的表,给其只读许可就完全足够了。

让最佳方法珠联璧合

除了上述的各种强安全措施，从 Web 应用程序的每一层中清除不用的代码、函数、库、内置的存储过程以及其它不重要的对象都是很好的方法。这要求深入理解每一层的工作机制，理解每一层需要哪些部分才能运作。然而，清除这些不必要的项目会极大地减少攻击面，从而减少了攻击者可用的工具。

SQL 注入一直居于其它 Web 应用程序漏洞之上。但是，通过使用上述多种技术的组合，你就可以阻止 SQL 注入攻击。记住，安全是一个动态的过程，你必须将安全性集成到系统开发生命周期的每一个步骤中。通过将安全性融合到开发生命周期中，并经常执行测试，你就可以赢得一个更好的企业安全环境。

在每一个程序投入使用之前，不妨考虑使用动态的 Web 应用程序测试，以及人工测试与源代码的检查来确保静态代码的安全性。

（作者：茫然　来源：TechTarget 中国）