

浅析 Web 应用软件开发安全

王青国

(江苏电力信息技术有限公司, 江苏 南京 210018)

摘 要: 本文结合国家电网公司信息化发展规划和现状, 剖析 Web 应用常见的安全漏洞, 分析漏洞的表现形式、形成原因、规避措施, 并提出了在软件开发生命周期全过程中预防安全漏洞的措施和方法。提出了从软件开发过程管理和技术手段两个方面系统性地减少 Web 应用安全风险的方法。本文旨在帮助广大 Web 应用开发人员了解常见安全漏洞、提高安全意识、掌握方法, 提高自身开发软件的安全性。文章内容对企业信息主管部门和应用软件开发单位加强安全管理、避免和减少安全事件的发生具有一定指导意义。

关键词: 信息安全; Web 应用; 安全漏洞; 安全风险; 安全测试; 代码审查; 代码扫描

0 引言

如今, 许多公司依赖基于 Web 的软件系统运营其业务流程、与供应商进行交易、向客户提供服务, 在每个应用程序上构建安全性应该是软件和系统交付业务流程的一部分。不幸的是, 为了保持竞争能力的领先地位, 许多公司不断地加快新产品的面市速度, 却忽略了应用系统的安全漏洞。导致黑客得以轻松访问或偷窃公司个人数据, 可能使整个公司处于风险之中^[1]。

随着企业信息化程度的不断提高, 作为企业信息化重要保障措施的信息安全引起业界的高度重视。近年来网络安全方面的技术研究和应用发展较快, 应用软件开发方面的安全技术研究比较少, 而权威统计数据表明 75% 以上的信息安全问题是由应用软件本身的安全漏洞引起的。本文结合电网企业信息化建设的现状, 分析常见 Web 应用安全风险, 提出在软件开发生命周期全过程中防范安全风险的措施和方法。

1 Web 应用安全的现状

1.1 重基础设施安全、轻应用软件安全

应用软件的安全问题一般分属于以下几个方面的原因: ①内因, 软件本身有安全问题。软件脆弱点(安全漏洞); 软件设计缺陷(设计层); 软件代码缺陷(实现层); 软件开发方法问题(开发过程)。②外因, 软件的运行环境。网络对软件的发展对软件安全产生了重大影响, 日益复杂的运行环境给攻击者提供了更多机会和方便, 而且攻击手段和方式

变化多端。③外部环境, 黑客、恶意代码等网络攻击行为增加了应用软件受到安全威胁的可能。④内部环境, 误操作、报复、经济犯罪等使得即使运行在企业内网的应用软件也面临严重的安全风险。

企业在防范安全风险的投入历来都是主要从外因和外部环境方面考虑, 重视基础设施方面的安全投入, 例如实施内外网隔离、建立了防火、设置入侵检测系统、部署网络安全工具等。以上措施的确可以从网络以及系统层面增强 Web 应用的安全性, 但却片面强调了硬件的作用而忽视了 Web 应用本身的安全问题。对于存在缺陷的应用程序来说, 再多的防护措施也将形同虚设。Web 安全是各种因素的综合体, 涵盖了网络、操作系统、应用服务器和 Web 应用本身的安全问题, 任何方面的缺失都会将应用暴露于黑客的攻击之下^[2]。Gartner 的报告表明发生在网络上的安全攻击大约 75% 针对 Web 应用, 约 67% 的 Web 程序存在安全漏洞^[5]。

1.2 Web2.0 增加了安全威胁的可能性

如今 Web2.0 技术已经在互联网应用和企业应用中广泛采用。Ajax、Flex 等新技术在改善了 Web 应用用户体验。同时给黑客攻击 Web 应用提供了更多可能性, 也是开发人员开发出安全的应用程序更加困难^[3]。

Web2.0 的典型技术 Ajax 具有动态改变内容和样式、异步通讯、非同源调用等技术特征, 这就为 XSS (跨站脚本攻击)、CSRF (跨站请求伪造)、DoS (拒绝服务攻击) 等攻击行为提供了更多的便利性。

越来越多的组织使用 Flex 技术开发 Flash 应用程

序,由于Flash支持全局变量,只要攻击者控制了全局变量就可以实施多种攻击行为。例如Cross-Site Flashing(跨站Flash攻击)、Cross-Site Scripting through Flash(通过Flash进行跨站脚本攻击)等^[3]。

首先,由于Web2.0新技术的引入,带来了新的安全隐患(如XSF攻击等);其次,很多Web1.0时代的安全隐患出现了新的传播渠道,如Ajax应用提高了XSS、CSRF、DoS等攻击的机会;最后,Web2.0时代应用程序的数量飞速增长、复杂度不断提高、开发部署速度加快,也给软件本身的质量和开发过程管理带来了困难,增加了出现安全漏洞的机会。

1.3 出现了安全风险控制前移的趋势

随着业界对Web应用安全风险重视程度的不断提高,安全风险控制技术逐渐从单纯的桌面、服务器、网络和网关等基础设置控制,向包括中间件和应用程序在内的整体解决方案转变。在软件生命周期的角度来看,逐步从重视运行维护阶段的监视和防护,向包括需求、设计、开发、测试、部署、维护在内的软件全生命周期安全控制和防范整体解决方案转变。由此也出现了一些新的方法论和工具软件,IBM、HP、微软等纷纷推出自己的完整安全解决方案和支撑工具软件。

例如,微软提出了安全软件开发模型和安全开发周期,OWASP提出了软件保证成熟模型。IBM提出了自己的安全解决方案,并发布了Rational AppScan等工具软件帮助开发人员解决安全问题^[3]。

2 常见Web应用安全风险分析

大多数的网络应用程序开发人员都不是安全方面的专家,也不会意识到大多数存在的安全问题和最佳实践。软件功能通常最受人重视,而安全性问题都会在最后才会得到处理。如果每一个应用程序开发人员都了解常见安全风险及应对措施库,无疑会开发出更加安全的应用程序。

本节旨在帮助开发人员和管理人员了解和正确处理一些Web应用程序常见的安全问题。能够影响Web应用安全的漏洞成百上千,但是大部分安全问题通常集中在几个方面。本文对OWASP(Open Web Application Security Project)发布的2010年Web应用十大安全风险进行分析,针对每个安全风险给出问题描述、预防措施,目的是培训开发人员、设计

人员、架构师、经理和企业组织,让他们认识到最严重的Web应用程序安全漏洞及产生的后果,并在自己的开发过程中避免这些问题。

2.1 注入攻击漏洞

不可信的数据作为命令或者查询语句的一部分被发送给解释器。攻击者发送的恶意数据可以欺骗解释器,以执行破坏性的命令或者访问未授权的数据。常见的注入漏洞包括SQL注入、OS注入、LDAP注入等。

需要将不可信数据从命令及查询中区分开。具体措施:①完全避免使用解释器而是使用安全的参数化API。但仍要注意有些参数化的API(比如存储过程),如果使用不当仍然可以引入注入漏洞。②如果无法使用参数化的API,那么应该使用转义语法来避免特殊字符。③使用恰当的验证方法对输入进行验证,由于很多应用允许在输入中有特殊字符,这一方法不是完整的防护方法。

2.2 跨站脚本攻击漏洞

应用程序收到含有不可信的数据,在没有进行适当验证和转义的情况下就将其发送给浏览器。攻击者可以利用XSS在受害者的浏览器上执行脚本,从而劫持会话、危害站点或者将用户转向恶意网站,从而发动安全攻击^[4]。

需要将不可信数据与动态的浏览器内容区分开。具体措施:①根据数据将要置于的HTML上下文(主题、JavaScript等)转义所有的不可信数据。②使用有恰当规范化和解码功能的输入验证方法对用户输入进行验证。由于很多应用允许在输入中有特殊字符,这一方法不是完整的防护方法。这种验证方法需要尽可能地解码任何编码输入,同时在接受输入之前需要充分验证数据的长度、字符、格式和业务规则。

2.3 失效的身份认证和会话管理

与身份认证和会话管理相关的应用程序功能往往得不到正确的实现,这就导致了攻击者破坏密码、密钥、会话令牌或者攻击其它的漏洞去冒充其他用户的身份^[4]。

主要措施:①采用一套统一、强大、成熟的认证和会话管理系统。这套系统应满足OWASP的应用程序安全验证标准。②做出最大努力避免跨站脚本漏洞,因这一漏洞可以用来盗窃用户会话令牌。

2.4 不安全的直接对象引用

当开发人员暴露一个对内部实现对象(文件、

目录、数据库密钥等)的引用时,就会产生一个不安全的直接对象引用。在没有访问控制检测或其它保护的情况下,攻击者就会操控或利用这些对象去访问未授权数据^[4]。

需要选择一个适当的方法来保护每一个用户可访问的对象以避免这类问题。主要措施:①使用基于用户或者会话的间接对象引用,这样能防止攻击者直接攻击未授权资源。②检查访问权限。任何来自不可信源的直接对象引用都必须通过访问控制检测,确保该用户对请求的对象有访问权限。

2.5 跨站请求伪造(CSRF)

迫使登录用户的浏览器将伪造的HTTP请求,包括该用户的会话cookie和其他认证信息,发送到一个存在漏洞的Web应用程序。这就允许了攻击者迫使用户浏览器向存在漏洞的应用程序发送请求,而这些请求会被应用程序认为是用户的合法请求^[4]。

要防止这类问题,需要在每个 HTTP 请求的主体或 URL 中添加一个不可预测的令牌(至少应该对每个会话是唯一的,也可以对每个请求是唯一的)。具体措施:①将独有的令牌包含在一个隐藏字段中,这将使得该令牌通过 HTTP 请求主体发送,避免其被包含在 URL 中从而被暴露出来。②将独有令牌包含在 URL 中或作为一个 URL 参数,但这样 URL 会暴露给攻击者。

2.6 安全配置错误

一般对应用程序、框架、应用服务器、Web服务器、数据库服务器定义和执行安全配置,以提高安全性。由于许多设置的默认值并不是安全的,因此就必须定义、实施和维护所有这些设置以确保安全性^[4]。

主要措施:①建立一个可以快速且易于部署在另一个环境的可重复的加固过程,并且尽量使这一过程自动化。开发、质量保证和生产环境都应该配置相同。②建立一个能及时了解并部署所有最新软件更新和补丁的过程,包括通常被忽略的所有代码的库文件。③建立一个能在组件之间提供良好的分离和安全性的强大应用程序架构。④实施漏洞扫描和经常进行审计以帮助检测可能的错误配置或没有安装的补丁。

2.7 不安全的加密存储

许多Web应用程序并没有使用恰当加密措施或Hash算法保护敏感数据,例如信用卡、社保卡、身份认证证书等。攻击者可能利用这些弱保护数据实

施身份盗窃、信用卡欺骗或其它犯罪^[4]。

对一些需要加密的敏感数据,应该起码做到以下几点:①预测一些威胁,加密这些数据的存储以确保免受这些威胁。②确保异地备份的数据被加密,并且用于加密的密钥和数据分开管理和备份。③确保使用了合适而强大的标准算法和强大的密钥,密钥管理到位。④确保使用强大的标准哈希算法来保护密码,确保所有密钥和密码都是被保护的且不被未经授权访问。

2.8 没有限制 URL 访问

许多应用程序在显示受保护的连接和按钮前会检测URL访问权限。但是,当这些页面被访问时却没有执行类似的访问控制检测,攻击者可以伪造这些URL去访问隐藏的网页^[4]。

要防止这类问题需要采用一种保护机制保证每个页面都需要适当的身份验证和适当的授权。通常,这种机制由应用程序代码之外的一个或多个组件提供。这种机制应该:①认证和授权策略应该基于角色,使维持这些策略的耗费最小化。②策略应该是高度可配置的,以尽量减少硬编码的策略问题。③在缺省情况下,应该拒绝所有访问,对于每个页面的访问,需要明确授予特定的用户和角色访问权限。④如果页面参与了工作流程,检查并确保当前的条件是授权访问此页面的合适状态。

2.9 传输层保护不足

应用程序时常没有进行身份认证、加密措施,甚至没有保护敏感网络数据的保密性和完整性。而当进行保护时,应用程序有时采用弱算法、使用过期或无效的证书、或不正确地使用这些技术,给攻击者可乘之机^[4]。

主要措施:①要求所有敏感网页都使用 SSL,对这些网页的非 SSL 请求被重定向到对应的 SSL 网页。②对所有敏感的 Cookie 都设置“安全”标志。③配置 SSL 的供应端只支持强大的标准(如 FIPS 140-2 标准)算法。④确保证书是有效的、没有过期、没有被废除,并匹配应用的使用的所有域。⑤后端和其它连接也应该使用 SSL 或其它加密技术。

2.10 未验证的重定向和转发

Web应用程序经常将用户重定向和转发到其它网站和页面,并且利用不可信的数据去判定目的页面。如果没有得到适当验证,攻击者可以重定向受害用户到钓鱼软件或恶意网站,或者使用转发去访问未授权的页面。钓鱼软件为了获取用户信任,往

往攻击这种漏洞^[4]。

主要措施包括：①避免使用重定向和转发。②如果使用了重定向和转发，则不要在计算目标时涉及到用户参数。③如果使用目标参数无法避免，应确保其所提供的值对于当前用户是有效的，并已经授权。

3 软件开发生命周期安全控制

3.1 软件开发周期安全防控概述

一方面，如上节所述我们必须研究应用的各个方面找出隐藏的漏洞。另一方面我们还必须采用一种方式检查软件开发生命周期的各个方面，从而将我们的应用软件构建得尽可能安全。必须将安全防控活动整合到软件开发生命周期各阶段。如图1所示。注意实际的软件开发过程不是严格按照以下瀑布模型顺序执行开发活动，而常常是以某种方式交替进行，安全防控活动也必须相应地交替执行。



图1 软件生命周期安全防控活动概述

3.2 各阶段的安全防控活动

3.2.1 需求阶段

1) 确定保护什么：首先要思考以下三个核心问题^[7]：①为什么要这样做？②在努力保护着什么？③如果没有做好它将会发生什么。以确定真正应该被保护的对象和目标。

2) 定义安全需求：遵照最佳实践定义明确的安全需求，保从一开始就将安全性构建到应用中。设定安全需求的目标是确保应用可以预防和抵挡攻击。典型的Web应用时安全需求包括：审计和日志记录，身份验证，会话管理，输入验证和输出编码，异常处理，加密要求，静止数据安全，活动数据安全，配置安全要求^[1]。

3.2.2 设计阶段

1) 安全架构分析：一般采用威胁建模技术进行安全架构分析。依次执行分解应用程序，确定面临威胁，威胁评估，确定缓解方案四个步骤^[3]。

2) 安全架构设计：安全地设计应用至关重要，

确保安全需求被正确地实现。一般包括缓解方案实现设计，安全框架选择或设计等内容。尽可能采用成熟的框架实现安全需求

3.2.3 开发阶段

大多数被报告的安全漏洞都是开发实践不佳的结果。开发人员必须假定用户将故意篡改公开的应用业务逻辑，并试图进行恶意攻击^[1]。最佳实践是将编码与测试并行执行，同时把相关工具整合到开发流程中，使开发过程中与安全性相关的许多任务自动化。

1) 源代码安全分析：扫描应用的源代码以查找漏洞和错误编码实践。例如，利用Rational AppScan Source Edition在开发环境中进行源代码分析，能够解决可能存在的大量安全性问题，减少在发布周期结束时可能出现的安全性测试瓶颈。这个活动需要在开发过程中持续执行。Rational AppScan Build Edition可将源代码分析自动化到构建过程中^[1]。

2) 动态安全分析：是一种渗透测试方法，试图自动扫描并记录攻击面，借助故障注入的手段来测试应用以及根据响应来确定漏洞的存在。例如，利用Rational AppScan Tester Edition在构建、测试过程中进行自动安全扫描分析，并将此活动延伸到交付阶段周期性地执行^[1]。

3.2.4 交付阶段

安全的软件如果被交付到不安全的环境中，也会不再安全。因此，需要在交付环境中最终审核应用安全性，然后维持操作环境的安全级别。主要措施包括应用基础架构的固化、数据通过网络时的保护、生产环境的防御以及配套操作系统和组件的修补和更新等。

1) 二进制安全分析：与源代码安全分析的操作类似，但只在二进制级别上分析，允许查找上下文风险和特定于平台的问题。一般在系统上线前由专门的团队执行，以评估应用系统的安全性^[3]。

2) 安全漏洞扫描：通过利用已知漏洞的大型数据库以及尝试识别应用中的已知漏洞来发现应用程序中存在的安全问题。例如，利用Rational AppScan Standard Edition在上线前执行安全评估并在运行维护阶段周期性的加以执行^[1]。

3) 安全监控分析：在系统的运行维护阶段需要持续对应用系统及环境进行监控，以及及时发现可能的安全问题并采取有效措施，一般由运行维护团队执行。

3.3 开发过程安全管理

要确保组织开发的 Web 应用软件安全,除了采取以上技术措施外,还应从软件开发过程管理的角度思考。建立适当的软件开发过程规范以将相关要求和活动整合到组织的软件开发过程中去。

对所有开发人员实施安全教育以提高安全意识,实施安全技术培训使之掌握常见的安全风险和规避措施。同时,还要确保开发、测试环境的信息安全,避免敏感测试数据、源代码等信息泄露并给攻击者利用。

4 结论

Web 应用安全存在一些已经典型的攻击模式和最佳实践,通过培训使所有开发人员认识风险的本质和预防措施,有利于开发出安全的应用程序。

在软件开发生命周期的各个阶段采取措施而不是在最后时刻发现和解决问题,把安全控制纳入日常的软件开发工作,可以大大降低应用程序出现安全问题的可能性。如何把安全管理的要求纳入软件开发规范并有效执行,是需要进一步研究的课题。

参考文献:

- [1] IBM. 利用 IBM Rational AppScan 改进 Web 应用软件开发生命周期的安全性. 2010 年 12 月.
- [2] Cesar E. Santiago, Maryann Hondo. Web 2.0 桌面与移动应用程序安全性设计. IBM developerWorks, 2012 年 1 月 4 日.
- [3] 赵静. Web 2.0 应用安全深入解析: 企业级 Web 2.0 应用安全解决方案. IBM developerWorks, 2009 年 8 月 27 日.
- [4] OWASP. OWASP Top 1-2010 The Ten Most Critical Web Application Security Risks, 2011 年 2 月.
- [5] 肖国一. Web 应用安全利器: IBM Rational AppScan. IBM developerWorks, 2010 年 2 月 25 日.
- [6] 程永敬, 翁海燕, 朱涛江, 等译. 编写安全的代码(第二版). 机械工业出版社, 2005 年 1 月.
- [7] Jack Danahy. 策划安全策略: 应询问的三个核心问题. IBM developerWorks, 2009 年 1 月 4 日.

作者简介:

王青国(1971—), 男, 内蒙古赤峰人, 高级工程师, 长期从事电力行业软件设计、开发、运行维护工作, 在 JavaEE 软件设计、开发方面有比较深入的研究,
E-mail: wang_qingguo@sohu.com。