Web 安全性测试技术综述*

于莉莉[1,2], 杜蒙杉[3], 张 平[1], 纪玲利[4]

(1. 北京航空航天大学 计算机科学与技术系, 北京 100191; 2. 二炮软件测评中心, 北京 100085; 3. 空军装备研究院 通信导航与指挥自动化研究所, 北京 100085; 4. 二炮装备研究院, 北京 100085)

摘 要: 对 Web 应用程序进行有效彻底的测试是及早发现安全漏洞、提高 Web 应用安全质量的一种重要手段。首先介绍了 Web 应用安全威胁分类,总结了常见的 Web 应用安全漏洞;然后对当前 Web 安全性测试技术的研究进行了全面概述,比较了静态技术和动态技术各自的优缺点,同时对在 Web 安全性测试中新兴涌现的模糊测试技术进行了详细的介绍和总结;最后指出了 Web 安全测试中有待解决的问题以及未来的研究方向。

关键词: Web 应用安全漏洞; 静态分析; 动态分析; 模糊测试

中图分类号: TP391 **文献标志码**: A **文章编号**: 1001-3695(2012)11-4001-05

doi:10.3969/j.issn.1001-3695.2012.11.001

Survey on Web security testing technologies

YU Li-li[1,2], DU Meng-shan[3], ZHANG Ping[1], JI Ling-li[4]

(1. *Dept. of Computer Science & Technology, Beihang University, Beijing* 100191, *China*; 2. *Software Testing & Evaluation Center of the Second Artillery Force, Beijing* 100085, *China*; 3. *Communication Navigation & Auto Command Institute, Air Force Equipment Academy, Beijing* 100085, *China*; 4. *The Second Artillery Force Equipment Academy, Beijing* 100085, *China*)

Abstract: A thorough test for Web application programs can be beneficial to finding security vulnerabilities promptly and improving the security quality of Web application. This paper firstly introduced the classification of Web application security threat and summarized common Web application security vulnerabilities. Then it comprehensively surveyed the state of Web security testing technology research, respectively compared weaknesses and merits of the static technologies and the dynamic technologies. At the same time, it introduced and concluded the detail of fuzzing, which was the new emerging technology in Web security testing. At last, this paper presented the problem to be solved and the future research direction.

Key words: Web application security vulnerabilities; static analysis; dynamic analysis; fuzzing

0 引言

近 20 年来,互联网技术在全球范围内得到了持续快速的发展。在我国的中国互联网络信息中心(CNNIC)2012 年 1 月发布的《中国互联网络发展状况统计报告》中指出:截止 2011 年 12 月,网民规模达到 5 亿,互联网普及率攀至 38.3%[1]。互联网已经成为一项重要的社会基础设施,Web 应用也逐渐成为软件开发的主流之一。以电子商务、社交网络、博客、论坛社区为代表的 Web 应用已经在人们的生活中扮演了越来越重要的角色,并逐渐改变了人们的生活方式。

但在 Web 应用中存在的各种安全漏洞也逐渐暴露出来,这些漏洞可能导致 Web 应用遭受各种攻击,严重影响了社会稳定、经济发展和人们的正常生活。根据 Symantec 发布的《Symantec Internet security threat report》[2],60% 以上的软件安全漏洞是关于 Web 应用的。这些安全漏洞可能会导致 Web 应用遭受各种攻击,如拒绝服务攻击、SQL 注入、窃取用户信息等。尽管防火墙、入侵检测等技术已经比较成熟,Web 应用的特殊性往往导致防范各类安全性问题的难度很大,因为 Web 应用攻击通常来自应用层,并且可能来自任何在线用户,甚至包括已经通过认证的用户[3]。开放式 Web 应用程序安全项目(open Web application security project,OWASP)在 2010 年公布的 Web 应用十大安全漏洞[4](图 1)中,排名靠前的注入(injection)、跨站点脚本(cross site scripting,XSS)、未能限制 URL 访问(failure to restrict URL access)等几种漏洞都是由于未能正确处理用户请求而引起的。

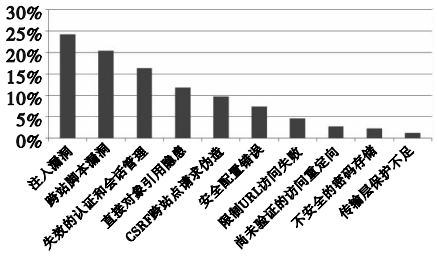

图1 OWASP十大安全漏洞(2010年)

造成 Web 应用安全漏洞的原因一方面是 Web 应用开发人员经验不足、对安全问题不够重视,更重要的则是缺乏全面彻底的安全性测试。根据 IEEE 的软件工程标准术语,软件测试被定义为:使用人工和自动手段来运行或测试某个系统的过程,其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别[5]。该定义中提到的需求通常包括了

收稿日期: 2012-03-25; **修回日期**: 2012-05-11 **基金项目**: 二炮研究院青年创新基金资助项目(2011619);国家自然科学基金资助项目(90718018)

作者简介:于莉莉(1978-),女,河南开封人,博士研究生,主要研究方向为回归测试、质性研究方法、软件维护(xueer-123@263.net);杜蒙杉(1978-),辽宁沈阳人,工程师,主要研究方向为软件工程;张平,工程师,主要研究方向为软件测试;纪琳俐,高级工程师,主要研究方向为软件安全.

软件功能、性能、可用性、安全性等诸多方面,而在传统 Web 应用测试过程中,测试人员更注重对正常功能的验证,往往忽视安全性方面的需求。Web 应用作为一种特殊的软件,面临着比传统单机软件更为严峻的安全威胁和更为复杂的用户环境。因此在 Web 应用发布前对其进行全面彻底的安全性测试,发掘 Web 应用中的安全漏洞和潜在的安全隐患是非常有必要的。

在 Web 应用领域中,发现安全漏洞的常用技术方法有静态分析技术和动态分析技术。模糊测试(fuzzing)技术作为软件测试研究中一种新的思路和方法,近年来开始被应用到 Web 应用测试领域。Web 应用存在的漏洞和安全隐患很大程度上是由于对用户的某些输入数据缺乏相应的校验或异常处理机制造成的[6]。在 Web 应用测试中,利用模糊测试技术有目的地构造大量的有效或者无效的用户输入数据提交给 Web 应用,同时能够通过多种方式监测 Web 应用的行为,进而分析任何引起 Web 应用出现异常甚至崩溃的原因,将有助于发掘 Web 应用中存在的安全漏洞和隐患,最终达到提高 Web 应用安全性的目的。

1 Web 应用安全漏洞

RFC 2828 将漏洞定义为系统设计、实现或者操作和管理中存在的缺陷和弱点,可能被攻击者所利用,从而突破系统的安全策略,访问未授权的资源和数据[7]。Web 应用安全漏洞则可以定义为一个 Web 系统的各个组件(包括 Web 应用、Web 服务器、数据库等)在设计与实现以及安全策略上的漏洞。

Web 应用安全协会(Web Application Security Consortium, WASC)[8]的 Web 安全威胁分类(threat classification,TC)项目将 Web 应用安全威胁分为如表 1 所示的六类。

表 1 Web 应用安全威胁分类

Web 安全威胁类别	概念
验证	用来确认某用户、服务或应用身份的攻击手段
授权	用来确认某用户、服务或应用身份的攻击手段
客户端攻击	用来扰乱或探测 Web 应用用户的攻击手段
命令执行	在 Web 服务器上执行远程命令的攻击手段
信息暴露	用来获取 Web 应用具体系统信息的攻击手段
逻辑性攻击	用来扰乱或探测 Web 应用逻辑流程的攻击手段

当前 Web 应用中常见的安全漏洞[8]主要为八类。

1)未被验证的输入

攻击者可以窜改 HTTP 请求的任何部分,包括 URL、查询字符串、HTTP 头部、Cookie、表单等,试图越过站点的安全机制。常见的输入窜改攻击方式有跨站点脚本、缓冲区溢出、格式化字符串攻击、SQL 注入等。

2)SQL 注入

SQL 注入是指攻击者在输入域中插入特殊字符,改变 SQL 查询的本意,欺骗数据库服务器执行非法操作,从而破坏数据库或非法获取存储在数据库中的数据清单的行为。SQL 注入是 Web 应用漏洞中最普遍也是最严重的漏洞之一。

3)跨站点脚本

跨站点脚本是指攻击者在客户端(通常是 Web 浏览器)通过 Web 页面提交的输入数据中嵌入恶意代码,若服务器将这些数据不经过滤或转义而直接返回,那么其他用户在访问该 Web 页面时这些恶意代码将在后者的客户端被执行,从而达到攻击者的恶意目的。

4)缓冲区溢出

攻击者可以利用这个漏洞发送特定请求给 Web 应用,使其执行任意代码。

5)隐藏的字段

用户可以通过在 Web 浏览器中执行“查看源文件”等操作查看这些字段的内容,然后手工修改这些字段的参数值,再通过 URL 参数传回给服务器端。攻击者可以通过修改 HTML 源文件中的这些隐藏字段达到恶意攻击的目的。

6)不恰当的异常处理

用户提交给 Web 应用的正常请求有可能频繁地产生错误和异常情况,如内存不足、系统调用失败、数据库链接错误等。不恰当的异常处理会将详细的内部错误信息如堆栈追踪(stack trace)、数据库结构以及错误代码等提供给攻击者,给 Web 应用带来安全隐患。

7)远程命令执行

Web 服务器可能简单地将用户提供的输入数据传递给其他应用或操作系统本身。如果这些输入数据没有经过适当的验证,那么攻击者就可以直接执行目标系统上的命令。

8)远程代码注入

引起这个漏洞的主要原因是 Web 应用开发者不良的编码习惯,如允许将未经验证的用户输入传递给如 include( )或 require( )这样的方法,这将允许本地应用或远程的 PHP 代码被包含进来,攻击者可将他们自己的 PHP 代码注入到目标 Web 应用中。

2 静态分析技术

静态分析(static analysis)技术是指在不执行的情况下对程序代码进行评估[9],是一种典型的白盒测试方法。其基本思想是通过分析程序的运行流程来构建程序工作的数学模型,然后对该数学模型进行审查以发掘程序的安全缺陷。常见的静态分析方法主要包括词法语法分析、模式匹配分析、数据流分析、补丁比较分析和模型化分析等。近年来,静态分析技术在 Web 应用安全性测试领域的研究成果主要包括以下内容。

Jovanovic 等人[10]实现了针对 PHP 语言的源代码静态分析工具 Pixy,该工具通过对 PHP 源代码执行静态数据流分析来发掘 PHP 应用中 SQL 注入、跨站点脚本等类型的漏洞,具有效率高、误报率低等优点,但是存在不能支持 PHP 的面向对象特性以及无法解决源代码中文件包含等问题。类似的工具还包括 PHP String Analyzer、PHP-Sat 等。

Huang 等人[11]将 Web 应用程序漏洞看做是一个安全信息流问题(secure information flow problem),提出了源自类型系统和类型状态的基于格的静态分析算法,并阐述了它的可靠性。同时,他们根据该算法实现了 WebSSARI 工具,并对 SourceForge. net 上 230 个开源 Web 应用进行了测试,发现其中 69 个有安全隐患。

Martin 等人[12]提出了一种目标导向的模型检验(goal-directed model-checking)方法来自动生成测试用例,发掘由 Java 开发的大型 Web 应用中基于污点数据(taint-based)的漏洞。他们选取了源代码总行数达到 130 000 行的 3 个大型 Web 应用进行实验,发现其中存在的 10 个 SQL 注入漏洞和 13 个跨站点脚本漏洞。

Chess 等人[9]从安全编程的角度详细介绍了如何利用静态分析技术改善软件的安全性,其中包括如何在 Web 应用安全性测试中应用静态分析技术去发掘 Web 应用在输入验证、会话管理方面的安全漏洞。

Stuttard 等人[13]将基于源代码审查的静态分析技术总结为以下三个步骤:a)从进入点开始追踪用户向 Web 应用提交的数据,审查负责处理这些数据的代码;b)在代码中搜索表示存在常见漏洞的签名,并审查这些签名,确定某个漏洞是否确实存在;c)对内在危险的代码进行逐行审查,理解应用程序的逻辑,并确定其中存在的所有问题,需要进行仔细审查的功能组件包括 Web 应用中的关键安全机制(身份验证、会话管理、访问控制与任何应用程序范围内的输入确认)、外部组件接口以及任何使用本地代码的情况。

静态分析技术具有以下几点优势:

a)具有程序内部代码的高度可视性,可以对程序进行全面分析,能够保证程序的所有执行路径得到检测,而不局限于特定的执行路径。

b)可以在执行前验证程序的安全性,进而在运行程序前检验程序安全性、修补安全漏洞。

c)无须实际执行被测程序,不会产生程序运行开销,快速高效,自动化程度高。

同时,静态分析技术也存在以下几点不足:

a)通用性较差,一般需要针对某种程序语言及其应用平台来设计特定的静态分析工具,具有一定的局限性。

b)误报(false positive)、漏报(false negative)率高,静态分析工具通常需要在两者之间寻求平衡。

c)静态分析技术依赖于被测程序的源代码。

3　动态分析技术

动态分析(dynamic analysis)技术是从外围或者客户端测试执行程序的技术,是一种典型的黑盒测试技术。面向 Web 应用的动态分析技术,比较具有代表性的是渗透测试(penetration testing)技术。渗透测试是指测试人员模拟恶意用户的行为对 Web 应用进行安全评估,针对 Web 应用中可能存在的代码缺陷、逻辑设计错误等问题进行测试,最终发掘其中的安全漏洞。开放式 Web 应用程序安全项目将 Web 应用渗透测试分为被动阶段(passive mode)和主动阶段(negative mode)来进行[14]。前者内测试人员需要尽可能地去搜集被测 Web 应用的信息,如通过使用 Web 代理观察 HTTP 请求和响应等,了解该应用的逻辑结构和所有的注入点(access point);后者内测试人员需要从各个角度、使用各种方法对被测 Web 应用进行渗透测试,主要包括配置管理测试、业务逻辑测试、认证测试、授权测试、会话管理测试、数据验证测试、拒绝服务测试、Web 服务测试和 AJAX 测试等。近年来,动态分析技术在 Web 应用安全性测试领域的研究成果主要有以下内容。

Ratproxy 是 Google 信息安全技术团队开发的被动式(passive)网络应用程序安全评估工具。运行 Ratproxy 后可以在本地启动一个代理服务器,测试人员需要将服务器地址设置为 Web 浏览器的代理,在后续测试中只需要使用浏览器正常地访问被测 Web 应用,如登录操作、填写表单等,Raproxy 会截获这些请求并对其进行窜改后发送给 Web 应用,并在后台记录相关日志,支持将测试报告导出为 HTML 格式。

由 Dafydd Stuttard 开发的 Burp Suite 是一款功能全面的 Web 应用渗透测试工具,Burp Suite 收费版本中提供了拦截代理服务器(burp proxy)、Web 爬虫(burp spider)、扫描器(burp intruder)、序列生成器(burp sequencer)、重放器(burp repeater)等部件。其中,拦截代理服务器部件提供了详细的拦截规则,能够准确地分析 HTTP 消息的结构和内容;Web 爬虫部件能够遍历被测 Web 应用,记录所访问的每个页面;扫描器部件可以自动实施各种定制攻击;序列生成器部件能够对会话令牌、会话标志或其他出于安全原因需要随机产生的键值的可预测性进行分析;重放器部件允许用户调整并重新提交之前被记录的请求。

Paros 是一款使用 Java 语言开发的 Web 安全漏洞扫描工具,同样支持对客户端和服务器之间的 HTTP/HTTPS 数据信息进行拦截和窜改,能够支持对 SQL 注入、跨站点脚本等多种类型漏洞的发掘。Paros 的优点在于它使用通过代理服务器的请求作为攻击的基础,能够有效地覆盖被测 Web 应用的所有功能。Nikto 是 Chris Sullo 使用 Perl 语言开发的开源 Web 安全黑盒测试工具,主要针对 Web 服务器的错误配置和不安全的文件和脚本进行测试,其底层功能基于 LibWhisker 实现,能够在 Windows、Linux 和 Mac OS 等多个平台上运行。Stuttard 等人在文献[13]中通过大量实例详细介绍了 Web 应用防御机制和常见的安全性问题,以及测试人员在渗透测试中使用的方法和技巧,包括对验证机制、会话管理、访问控制的攻击等。Hope 等人在文献[15]中从工程实践的角度较为系统地介绍了 Web 应用安全测试中常用的动态分析技术,包括以多种方式窜改输入、自动化批量扫描、操纵会话等。

与静态分析技术相比,动态分析技术通常是在真实的运行环境中对被测目标进行测试,直接模拟攻击者的行为,因此其测试结果往往具有更高的准确性,漏报、误报率也相对较低;此外,动态分析技术不需要应用程序的源代码,只需提供与被测 Web 应用的网络连接即可,具有较高的灵活性。

4　模糊测试技术

Fuzzing 技术源于软件测试中的黑盒测试技术,其基本思想是自动产生和发送大量随机或经过变异的输入值给软件,一旦发生失效或异常,便可挖掘出软件系统存在的薄弱环节和安全漏洞[15~21]。相比传统的黑盒测试方法,模糊测试关注任何可能引发未定义或者不安全行为的输入,其优点包括简单、有效、自动化程度高以及可复用性强等;其缺点也十分明显,如不可避免地生成大量冗余的测试数据、代码覆盖率不足以及黑盒测试共有的低智能性。低智能性即黑盒测试通常只对程序的初始状态进行测试,而很多程序的缺陷都是隐藏在程序的后续状态中的。

4.1　模糊测试技术在 Web 应用测试中的发展

面向 Web 应用的模糊测试技术的基本思想是模拟恶意用户行为,有目的地构造大量异常、非法、包含攻击载荷的模糊测试数据并提交给 Web 应用,同时监测 Web 应用的行为,判断 Web 应用中是否存在安全漏洞。近年来,面向 Web 应用的模糊测试技术研究中涌现出了许多研究成果,其中一部分已经在 Web 应用的测试实践中得到了广泛的应用和认可。Hammersland[22]认为用户输入可能会使 Web 应用处于一种不确定状态,从而导致 Web 应用出现异常行为。他在研究中提出用一

种半自动化的模糊测试方法来测试Web应用处理各种类型用户输入的能力。Hammersland的研究主要关注Web表单这种类型的数据输入方式,所实现的模糊测试工具如图2所示。他选取了八款流行的开源Web应用作为测试对象进行实验,测试结果显示其中五款Web应用中均存在不同程度的问题,如针对某些特定类型的用户输入会导致WordPress应用出现崩溃等。

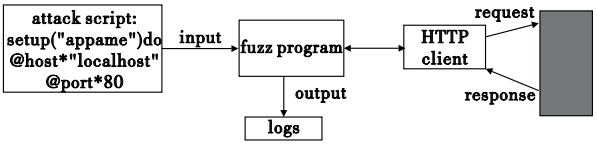

图2 Hammersland实现的模糊测试工具

Graaf[24]认为Hammersland的研究[23]中采用的模糊测试方法是非智能化的,并在此基础上提出了一种更加智能化模糊测试方法。他在研究中针对使用特定Web框架的Web应用生成半合法的模糊数据以避免其被Web框架所带的输入验证模块过滤或拒绝。WebScarab[25]是OWASP开发的一款全面的Web应用安全测试工具,它包含了一个基本的模糊器,可以向Web应用的参数注入模糊值。WSFuzzer[26]同样是由OWASP组织开发的一款模糊测试工具,可以自动实现Web服务通信中参数的模糊化,用户只需提供相应的Web服务描述语言(Web services description language,WSDL)文件或者定义服务SOAP结构的XML文件。SPIKE Proxy[27]是由Dave Aitel用Python语言开发的、基于浏览器的Web模糊测试器。它通过捕获Web浏览器请求,然后允许用户为一个目标Web应用运行一系列预定义的审核,以识别SQL注入、缓冲区溢出和跨站点脚本等类型的漏洞。WebFuzz是一款基于Windows平台的图形化界面的模糊测试工具,它能够有效地检测Web应用中存在的目录遍历、缓冲区溢出、SQL注入等类型的漏洞,它要求测试人员对HTTP有较深的理解以生成有效的测试数据。

在商业Web应用模糊测试工具方面,目前较为流行的主要有以下几款:SPI Dynamics公司开发的Web应用测试工具WebInspect中以构件的形式提供了图形化的SPI模糊器[28],SPI模糊器向用户提供了对模糊测试所使用的原始HTTP请求的完整控制,它要求用户具备基本的HTTP知识以开发测试用例并生成结果,同时也需要使用这些知识来解释Web应用的响应以识别那些需要进一步研究的问题;Beyond Security开发的beSTORM工具[29]可以为包括HTTP在内的多种协议产生模糊测试集,与之类似的还包括Codenomicon开发的HTTP测试工具[30];HP推出的下一代Web应用安全性测试工具集WebInspect中附带的一款具有图形化界面的模糊测试工具[31],能够对Web应用以及Web服务器进行模糊测试,实用性较强;SDL Regez Fuzzer[32]是Microsoft最近发布的一款针对Web应用安全的模糊测试工具,该工具能够帮助Web开发和测试人员发掘Web应用中容易接受拒绝服务攻击的正则表达式,Web攻击者可能会利用正则表达式中执行起来极为耗时的某些条款(如分组条款中包含自身的重复)来发起拒绝服务攻击。

本文对WebScarab、WebFuzz、JBroFuzz等九款开源或商业Web应用模糊测试工具从自动化程度、可扩展性等方面进行了对比,结果如表2所示。

目前,面向Web应用的模糊测试技术研究在国内开展得还比较少,主要研究成果包括:文献[33]中研究了基于模糊测试方法的Web应用的安全性测试技术,针对常见的漏洞类型设计了具体的模糊测试方案,并在此基础上开发了图形化的Web应用模糊测试工具WFT;文献[34]针对跨站点脚本漏洞,基于模糊测试技术设计了一种新的漏洞检测模型,该模型具有实施方便、自动化程序高等优点,提出一种基于服务器过滤删除机制的上下文无关文法模糊数据动态生成算法,能够对未知漏洞进行有效检测。

表2 Web应用模糊测试工具对比

工具名称	开发语言	可视化界面	针对的漏洞类型	自动化程度	可扩展性
WebScarab	Java	有	单一	一般	差
WSFuzzer	Python	无	单一	低	差
WebFuzz	C#	有	全面	一般	差
Wfuzz	Python	无	单一	低	好
SPI模糊器	C#	有	全面	一般	差
JBroFuzz	Java	有	全面	低	一般
RatProxy	C	无	单一	一般	差
ProxyStrike	Python	有	单一	低	一般
WFT	C#	有	单一	低	一般

4.2 存在的不足

尽管有大量研究成果和工程实践证明模糊技术可以应用到Web应用安全性测试实践中,但不可否认的是,当前的研究中还存在许多未解决或者待完善的问题,主要包括以下几个方面:

a)测试自动化程度低。大部分方法或工具在模糊数据的生成以及对被测对象监测结果的分析等过程都需要人工参与,自动化程度不高。例如Wfuzz、JBroFuzz等工具需要测试人员提供正常请求并对其中需要模糊化的变量进行标记才能生成一系列模糊数据。

b)支持的漏洞类型较少。部分工具只支持对几种特定类型的安全漏洞执行模糊测试,导致漏洞发掘能力有限,如WebFuzz、WFT等工具只能够发掘目标Web应用中SQL注入和跨站点脚本类型的漏洞。

c)漏洞检测的漏报、误报率高。部分工具所采用的模糊数据生成以及漏洞检测方法较为简单,造成测试结果中漏洞的漏报、误报率高。例如WebFuzz、ProxyStrike等工具只是通过在原始请求中简单地插入攻击载荷的方式来生成模糊数据,漏洞检测上也只是简单地通过查找返回的Web页面中是否存在特定的内容。

d)工具的可扩展性较差。WebFuzz等工具在设计上均存在耦合程度高、可扩展性差的问题,测试人员在其基础上增加对新漏洞类型的支持的代价较大。

e)测试结果的展示不够直观。大部分工具在测试结果的展示上都不够直观,有的甚至仅提供模糊测试的执行日志,如WSFuzzer、Wfuzz等,测试人员经常需要手动地对数百条记录进行分析来获取其中的哪些引发了被测对象的安全漏洞。

5 结束语

安全漏洞是研究安全问题的生命线。造成Web应用包含各类安全漏洞和安全隐患的原因很多,其主要因素包括不成熟的安全意识、开发/测试人员经验不足、Web应用越来越复杂、

资源和时间的限制、迅速发展的形势带来的威胁等。对于持续快速发展的 Web 应用而言,存在的安全漏洞在很大程度上都是由于 Web 应用自身对用户的输入数据缺乏有效的校验和过滤所造成的。因此,针对不同的应用情况,采取合适的处理方式,执行有效的安全测试是十分必要的。

鉴于上述目的,本文依据对 Web 应用程序代码处理的不同方式,分别介绍了三种发现 Web 应用安全漏洞的技术,即静态分析、动态分析和模糊测试,并依次分析了三种技术的优缺点。针对传统的静态分析和动态分析两种方法的优缺点,建议采取相互结合的方式以达到取长补短、提高查找安全漏洞能力的目的。而针对目前模糊测试技术在 Web 应用中存在的自动化程度不高、漏报/误报率高、可扩展性差等不足,结合目前开展的研究工作,笔者认为应该从自动分析 Web 应用、改进漏洞发掘方法、提高可扩展性等方面入手。

参考文献:

[1] CNNIC. 中国互联网发展状况统计报告(2012 年 1 月)[R/OL]. (2012-01-16) http://www. cnnic. net. cn/hlfzyj/hlwxzbg/201201/P020120709345264469680. pdf.

[2] Symantec Corporation. Symatec Internet security threat report. trends for January-June 07, Volume XII[R]. 2007.

[3] TIPTON H F, KRAUSE M. Information security management handbook[M]. 6th ed. New York: Auerbach Publictions, 2006.

[4] Open Web Application Security Project. OWASP top 10-2010: the ten most critical Web application security risks[R]. 2010.

[5] IEEE STD 610. 12-1990. IEEE standard glossary of software engineering terminology[S]. New York: IEEE, 1990.

[6] HUANG Yao-wen, HUANG S K, LIN T P, *et al*. Web application security assessment by fault injection and behavior monitoring [C]// Proc of the 12th International World Wide Web Conference. New York: ACM Press, 2003: 148-159.

[7] SHIREY R. RFC 2828, Internet security glossay[S]. 2000.

[8] ZHANG Jin, DIMITROFF A. The impact of metadata implementation on Webpage visibility in search engine results (part II)[J]. Information Processing and Management, 2005, 41(3): 691-715.

[9] CHESS B, WEST J. 安全编程: 代码静态分析[M]. 董启雄,韩年,译. 北京: 机械工业出版社, 2008.

[10] JOVANOVIC N, KRUEGEL C, KIRDA E. Pixy: a static analysis tool for detecting Web application culnerabilities (short paper)[C]//Proc of IEEE Symposium on Security and Privacy. Washington DC: IEEE Computer Society, 2006: 258-263.

[11] HUANG Yai-wen, YU Fang, HANG C, *et al*. Securing Web application code by static analysis and runtime protection [C]//Proc of the 13th International World Wide Web Conference. New York: ACM Press, 2004: 40-52.

[12] MARTIN M, LAM M. Automatic generation of XSS and SQL injection attacks with goal-directed model checking[C]//Proc of the 17th USENIX Security Symposium. Berkeley: USENIX Association, 2008: 31-43.

[13] STUTTARD D, PINTO M. 黑客攻防技术宝典: Web 实战篇[M]. 石华耀,等译. 北京: 人民邮电出版社, 2009.

[14] Open Web Application Security Project. OWASP testing guide v3.0 [R/OL]. (2008). https://www. owasp. org/images/5/56/OWASP_Testing_Guide_v3. pdf.

[15] HOPE P, WALTBER B. Web 安全测试[M]. 傅鑫,等译. 北京: 清华大学出版社, 2010.

[16] OEHLERT P. Violating assumptions with fuzzing[J]. IEEE Security and Privacy, 2005, 3(2): 58-62.

[17] MILLER B P, FREDRIKSEN L, SO B. An empirical study of the reliability of UNIX utilities[J]. Communications of the ACM, 1990, 33(12): 32-44.

[18] LANZI A, MARTIGNONI L, MONGA M, *et al*. A smart fuzzer for x86 executables[C]//Proc of the 3rd International Workshop on Software Engineering for Secure Systems. Washington DC: IEEE Computer Society, 2007.

[19] VUAGNOUX M. Autodafé: an act of software torture[R]. [S. l.]: Swiss Federal Institute of Technology, Cryptograhy and Security Laboratory, 2006.

[20] SUTTON M, GREENE A, AMINI P. 模糊测试:强制性安全漏洞发掘[M]. 黄陇,于莉莉,李虎,译. 北京:机械工业出版社,2009.

[21] 吴志勇,王红川,孙乐昌,等. Fuzzing 技术综述[J]. 计算机应用研究,2010,27(3):829-832.

[22] HAMMERSLAND R. Finding weaknesses in Web applications through the means of fuzzing [D]. Gjφvik, Norway: Gjφvik University, 2008.

[23] HAMMERSLAND R, SNEKKENES E. Fuzz testing of Web applications[EB/OL]. http://rune. hammersland. net/tekst/fuzzing_article. pdf.

[24] De GRAAF M. Intelligent fuzzing of Web applications[D]. Amsterdam: Universit of Amsterdam, 2009.

[25] OWASO. Category: OWASO Web Scarab project[EB/OL]. (2012-01-08). https://www. owasp. org/index. php/category: OWASO-Webscarab_project.

[26] WADDELL I, JONES N, STEED C, *et al*. Using the workflow technology in secure software engineering education[C]//Proc of the 14th Colloquium for Information Systems Security Education. 2010: 76-82.

[27] STOECKEL D M, STELZER E A, DICK L K. Evaluation of two spike-and-recovery controls for assessment of extraction efficiency in microbial source tracking studies[J]. Water Research, 2009, 43(19): 4820-4827.

[28] LIN Jin-cheng, CHEN Jan-min. An automated mechanism for secure input handling[J]. Journal of Computers, 2009, 4(9): 837-844.

[29] ERIKSSON B I, DAHL O E, BÜLLER R, *et al*. A new oral direct thrombin inhibitor, dabigatran etexilate, compared with enoxaparin for prevention of thromboembolic events following total hip or knee replacement: the BISTRO II randomized trial[J]. Journal of Thrombosis and Haemostasis, 2005, 3(1): 103-111.

[30] HENTEHZADEH N, MEHTA A, GURBANI N K, *et al*. Statistical analysis of self-similar session initiation protocol (SIP) messages for anomaly detection[C]//Proc of the 4th International Conference on New Technologies, Mobility and Security. 2011: 1-5.

[31] SPI Dynamics. Web application security assessment[R]. 2003.

[32] AMMANN P, WIJESEKERA D, KAUSHIK S. Scalable, graph-based network vulnerability analysis[C]//Proc of the 9th ACM Conference on Computer and Communications Security. New York: ACM Press, 2002: 217-224.

[33] 都娟. 基于模糊测试方法的 Web 应用安全性测试技术的研究及其工具实现[D]. 上海:华东师范大学,2011.

[34] 刘为. 基于模糊测试的 XSS 漏洞检测系统研究与实现[D]. 长沙:湖南大学,2010.