基于行为语义分析的Web恶意代码检测机制研究

李道丰[1,2] 黄凡玲[3] 刘水祥[4] 黄安妮[1,2]
(广西大学计算机与电子信息学院 南宁 530004)[1]
(广西高校并行分布式计算技术重点实验室 南宁 530004)[2]
(清华大学软件学院 北京 100084)[3] (中南大学软件学院 长沙 410075)[4]

摘 要 在Web安全问题的研究中,如何提高Web恶意代码的检测效率一直是Web恶意代码检测方法研究中需要解决的问题。为此,针对跨站脚本漏洞、ActiveX控件漏洞和Web Shellcode方面的检测,提出一种基于行为语义分析的Web恶意代码检测机制。通过对上述漏洞的行为和语义进行分析,提取行为特征,构建Web客户端脚本解析引擎和Web Shellcode检测引擎,实现对跨站脚本漏洞、ActiveX控件漏洞和Web Shellcode等的正确检测,以及对Web Shellcode攻击行为进行取证的功能。实验分析结果表明,新的Web恶意代码检测机制具有检测能力强、漏检率低的性能。

关键词 网页恶意代码,客户端攻击,检测,行为语义分析
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.8.023

Study on Web-based Malware Detection Mechanism Based on Behavior and Semantic Analysis

LI Dao-feng[1,2] HUANG Fan-ling[3] LIU Shui-xiang[4] HUANG An-ni[1,2]
(School of Computer and Electronic Information, Guangxi University, Nanning 530004, China)[1]
(Guangxi Colleges and University Key Laboratory of Parallel and Distributed Computing Technology, Nanning 530004, China)[2]
(School of Software, Tsinghua University, Beijing 100084, China)[3]
(School of Software, Central South University, Changsha 410075, China)[4]

Abstract How to improve the detection efficiency of Web malicious code has always been a problem to be solved in the research of Web security issues. A detection mechanism for Web-based malware based on behavior and semantic analysis was proposed to detect vulnerabilities in XSS, ActiveX controls and Web Shellcode in our paper. Behavioral characteristics was extracted and the detection engine was built to realize the correct detection of vulnerabilities in XSS, ActiveX controls and Web Shellcode, and the forensics of Shellcode attacks. Experimental results show that the performance of the new detection mechanism for Web-based malware has stronger detection ability and lower missing rate.

Keywords Web-based malware, Client attack, Detection, Behavior and semantic analysis

Web应用逐渐推广和普及,安全问题也随之而生[1,2],如SQL注入、跨站脚本攻击以及文件上传漏洞攻击等。攻击者通过这些漏洞来渗透网站并拿下Webshell,同时在网站挂上具有破坏或窃取信息行为的恶意代码(攻击Shellcode或网页木马),以Web服务器作为跳板,渗入内网,窃取内网敏感数据等。Web恶意代码主要通过客户端浏览器或其插件的漏洞触发并自动成功下载和运行,其攻击直接从客户端发起,易于避开安全防御设备的检测和传播。同时,由于基于Web的应用程序得到广泛应用和普及,因此Web恶意代码的攻击方式是目前比较流行的攻击手法。正因为这样,Web恶意代码的检测方法研究是备受安全研究者关注的Web安全研究主题之一。

Web恶意代码检测方法主要包括静态分析的检测方法和动态分析的检测方法,以及这两种方法的有机结合,具体方法参考文献[3-12]。其中,静态分析检测方法主要包括基于规则和特征码匹配方式[5,10]以及基于网页中的静态特征匹配方式来进行检测[8],但该方法仍存在漏检率高的不足;动态分析检测方法主要包括基于客户端蜜罐[4,5,8,11]和基于模拟仿真的检测方法[4,12]来进行检测。但目前这些方法仍存在一定的误判和漏检。为此,研究新的具有低误报率和漏检率的Web恶意代码检测机制具有重要的实用价值。

文中在研究Web恶意代码检测方法的过程中,通过恶意代码的静态特征即语义进行初步的定位,提出基于行为语义分析的Web恶意代码检测机制,侧重于对跨站脚本漏洞、ActiveX控件漏洞以及Shellcode的行为特征进行深入的研究并实现较为完备的检测方案。进一步采用动态模拟的方法跟踪

到稿日期:2015-07-14 返修日期:2015-11-09 本文受国家自然科学基金(61362010),广西自然科学基金项目(2012GXNSFAA053217),广西教育厅科研基金(YB2014008),广西研究生教育创新计划资助项目(YCSZ2015035)资助。
李道丰(1974-),男,博士,副教授,硕士生导师,主要研究方向为密码学与信息安全,E-mail:ldf_0123@163.com;黄凡玲(1992-),硕士生,主要研究方向为信息安全;刘水祥(1993-),男,硕士生,主要研究方向为信息安全;黄安妮(1991-),女,硕士生,主要研究方向为信息安全。

代码执行路径(行为),进一步确定是否存在威胁并给以正确判别,提高检测的准确率并降低漏检率。同时,在所提检测机制中,若发现恶意代码,在对其进行动态模拟执行的过程中捕获其攻击行为路径,以便提供取证工作。

1 预备知识

本节给出与本文工作相关的定义和概念。

定义 1 行为语义分析是指用户和服务提供者之间在交互过程中所表现的行为及其类型的审查,其中用户和服务提供者包括程序的客户端和服务端等。

本文所提到的语义分析是指根据代码特点以及相关语言的语法特点,定位页面中可能存在威胁的位置。

例如:利用 iframe 标签进行框架挂马,通过框架挂马结构:

〈iframe src=地址 width=0 height=0〉〈/iframe〉

攻击者常常会在 iframe 标签的 src 属性中加入外链、脚本、可执行文件,可以根据语法特点将可能存在威胁的位置定位到该行代码中。用类似的方法可以定位其它类似的恶意代码点。

行为分析是指通过仿真模拟的环境来执行可能存在恶意威胁的页面,追踪其执行过程并且采集相关的行为特征和行为路径,进行行为检测。若发现其行为异常,则发出警报。就上述例子而言,应进一步对 iframe 标签中的 src 属性值进行判断,看其是否为脚本或可执行文件。

通常地,由于 Web 恶意代码躲避检测技术的不断改进和提高,为了更全面和准确地检测 Web 恶意代码,不仅需要静态分析来判断是否为恶意代码,而且还需要通过 Javascript 引擎进行动态执行,并根据其行为特征判断其是否存在威胁。

2 新的 Web 恶意代码检测机制

通过对多种流行客户端攻击方式以及检查原理的研究,建立基于行为语义分析方法的恶意代码检测模型,并且实现 Web 恶意代码自动检测系统。如图 1 所示,新检测机制的运行过程如下:

(1)利用爬虫技术抓取页面,提取静态元素并进行语义分析;

(2)模拟执行可能出现的行为并进行分析,探测行为预期结果;

(3)模拟执行完成后,可能会转入/调用新的(或原来的)静态元素,重复步骤(1)和(2);

(4)对发生的所有行为过程提取行为执行路径,进行检测识别;

(5)输出检测结果。

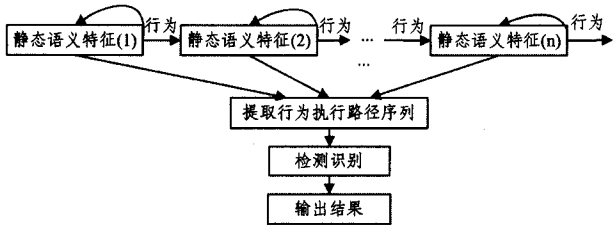

图 1 Web 恶意代码检测机制

以上检测机制的实现过程中,主要通过以下两个模块来实现。

Web 客户端脚本解析引擎:主要完成对跨站攻击、ActiveX 插件挂马等 Web 恶意代码的检测。

Shellcode 检测引擎:主要完成对 Shellcode 包含的 3 种常见的 GetPC code:jmp/call/pop、浮点指令/fstenv/pop、SHE 和 Kernel32.dll base address resolution 的检测。

2.1 Web 恶意代码检测

主要针对 Web 应用中存在的跨站脚本漏洞和 ActiveX 控件漏洞进行检测。具体实现如图 2 所示。

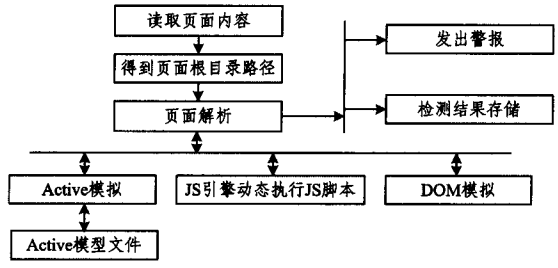

图 2 Web 恶意代码检测的实现

在 Web 恶意代码检测机制中,先读取 Web 站点上的页面内容以及页面所在的根目录路径,并对读取的页面内容进行解析,然后协同基于 DOM 树、ActiveX 对象以及 JavaScript 等检测引擎,根据我们的行为分析模型,完成对跨站脚本漏洞和 ActiveX 控件漏洞进行的检测。

其中,对于反射型跨站脚本漏洞和基于 DOM 的跨站脚本漏洞的检测,提取输入点 URL 作为行为特征,并构造 3 种正则表达式作为查找输入点的检测模型,然后利用模式匹配方法来进行检测,取得了很好的检测效果。3 种正则表达式的构造如下:

(1)匹配大部分 DOMXSS 源的正则表达式

/((src|href|data|location|code|value|action)\s*["'\]]*\s*\+?\s*=)|((replace|assign|navigate|getResponseHeader|open(Dialog)?|showModalDialog|eval|evaluate|execCommand|execScript|setTimeout|setInterval)\s*["'\]]*\s*\()/

(2)匹配大部分 DOMXSS 输出方式的正则表达式

/(location\s*[\[.])|([.\[]\s*["']?\s*(arguments|dialogArguments|innerHTML|write(ln)?|open(Dialog)?|showModalDialog|cookie|URL|documentURI|baseURI|referrer|name|opener|parent|top|content|self|frames)\W)|(localStorage|sessionStorage|Database)/

(3)匹配基于 jQuery 的输出方式的正则表达式

/after\(|\.append\(|\.before\(|\.html\(|\.prepend\(|\.replaceWith\(|\.wrap\(|\.wrapAll\(|\$\(|\.globalEval\(|\.add\(|jQUery\(|\$\(|\.parseHTML\(/

2.2 Web 脚本中的 Shellcode 检测

Shellcode 是一段需要通过某种漏洞来触发执行攻击的代码。为了逃过检测和查杀,Shellcode 一般采用多态技术如加密技术来掩护自己。但 Shellcode 在运行时,仍呈现一定的行为特征。在 Windows 系统中,变型后的 Shellcode 在执行过程中通常需要先自定位,也就是先自身解密,得到解密后的

shellcode 所在的内存地址；然后再获取 kernel32. dll 地址。目前 Shellcode 自定位的方法有 CALL GetPC、FSTENV GetPC 和 SEH GetPC。结合 Shellcode 自定位行为和获取的 kernel32. dll 地址行为，给出 Web 脚本中 Shellcode 的检测机制，实现对 GetPCode：jmp/call/pop、浮点指令/fstenv/pop、SHE 和 Kernel32. dll base address resolution 的行为检测，具体操作如图 3 所示。

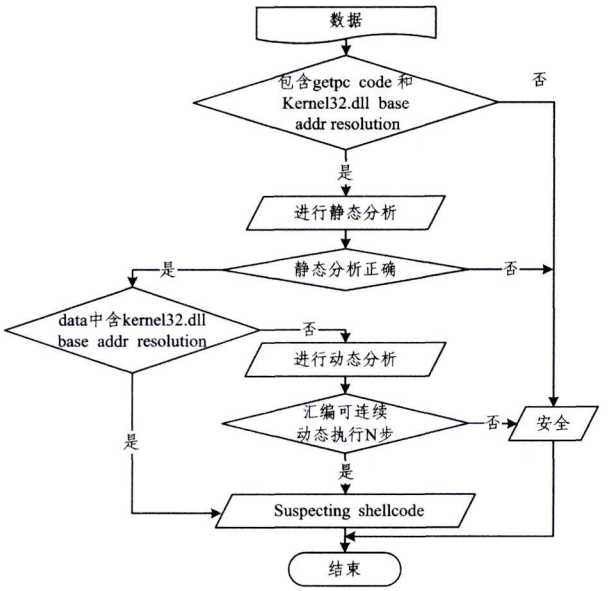

图 3　Web 中 Shellcode 检测的实现

如图 3 所示，先对程序进行静态分析，检测程序数据是否含有 getpc code 和 kernel32. dll base address resolution 指令。若没有，则该程序不是 Shellcode；否则，对程序进行动态分析，检测程序行为，并对行为特征进行分析，给出检测结果。在检测机制中，新增加对 FnstenvGetpc code、SHE getpc code、Kernel32. dll base address resolution 等类型行为的检测，由于增加了检测类型并扩大了检测范围，因此能更准确地检测不同类型的 Shellcode。

3　实验分析

本节从检测准确性和范围来评估检测机制的性能，也就是检测机制能正确识别网络中恶意代码的准确程度及其类型的程度。因此对机制性能进行评估时，主要从 2 个方面进行考查：1）识别恶意代码的准确性，即根据所提出的行为分析和检测模型，对被检测的页面和程序进行检测和识别的准确程度；2）恶意代码检测的类型和范围，即结合闭合标签新类型，提取相关行为特征并加以分析，检测更多类型的 Shellcode，以降低漏检率。

3.1　实验步骤

为了便于考查检测机制的检测准确性和检测类型范围，实验中，考虑在开源的 PhonyC 项目中实现所提出的检测机制，定制新的 Web 客户端脚本解析引擎和 Shellcode 检测引擎（ExLibemu），并给出新的原型系统即 WebMCDS。

实验的具体步骤如下：

（1）收集被检测 Web 站点及相关页面。对乌云网或 exploit-db 等安全网站所公布的网站以及互联网上的部分 Web 站点进行收集和筛选。

（2）生成新的测试样本。对已收集的 Web 站点及其程序相关的恶意代码样本，利用恶意代码多态引擎来生成新型的恶意代码变种。

（3）检测正确性测试。利用所给的检测机制对收集的 Web 站点进行检测，包括 Web 漏洞检测和多态的 Shellcode 检测以及取证功能。

（4）检测性能测试。针对不同类型的恶意代码进行检测，以判别检测机制识别恶意代码的类型范围、误报率以及漏检率。

3.2　结果分析

下面对根据 3.1 节中的实验步骤得到的结果进行分析。在正确性分析中，利用检测机制对给定包含 Shellcode 的样本网站进行检测，并对检测结果进行分析，检验检测机制识别恶意代码的准确性。同时增加包含不同类型 Shellcode 的样本，以检验检测模型的识别能力。

在实验中，检测样本网页数量为 10000 个，包含不同类型 Shellcode 的样本共有 1368 个，包含 FnstenvGetpc code 类型 546 个、SHE getpc code 类型 203 个、Kernel32. dll base address resolution 类型 327 个，其他类型 292 个。

（1）正确性

在 3.1 节实验步骤（3）中，利用 WebMCDS 原型系统对 10000 个 Web 脚本进行检测，能正确检测出 1368 个不同行为类型 Shellcode 的样本，并将结果记录于表 1 中，检测正确率 100%，误报率为 1.35%，其中一个包含 Web Shellcode 的检测结果如图 4 所示。

图 4　Web Shellcode 检测结果

从图 4 中可知，被检测页面包含 Shellcode 十六进制数据和 Shellcode 执行过程中所调用的函数以及函数的相关信息。同时还可以得到 Shellcode 攻击行为路径，亦即 Shellcode 执行了创建用户并提升管理员权限的非法攻击行为，可作为 Shellcode 攻击的取证结果。

而且，WebMCDS 系统将分析检测后的结果存入 SQLite 数据库中，用来表示页面之间的调用层次关系，记录数据格式如表 1 所列。

表 1　检测结果记录

rootUrl	maliciousUrl	layer	alertFlag
file://test8. html	file://shellcode2. html	1	1
file://test8. html	file://shellcode2. html	1	1
file://test8. html	file://shellcode2. html	1	1
file://test8. html	file://shellcode2. html	1	1

表 1 中 rootUrl 代表父链接，maliciousUrl 代表包含 Web 恶意代码的页面链接，layer 代表从 rootUrl 调用 maliciousUrl 所经过的层数。

(2)检测性能

1)Web 恶意代码类型检测能力

根据 3.1 节实验步骤(4),主要从 Web 恶意代码的检测范围来考虑 WebMCDS 的检测能力,包括 Web 漏洞类型和 Shellcode 行为类型的检测能力。

在 Web 漏洞类型的检测方面,WebMCDS 能检测 ActiveX 漏洞和跨站脚本漏洞中的反射型和 DOM 型漏洞,但不能检测存储型的漏洞,如表 2 所列。

表 2 Web 客户端脚本解析引擎的检测能力

性能指标	ActiveX 漏洞	跨站脚本漏洞		
		反射型	DOM 型	存储型
识别能力	√	√	√	×

其中,符号√表示对相应的漏洞具有一定的检测能力;×表示不具备该类型的检测能力。

在 Shellcode 行为类型的检测方面,表 3 列出 ExLibemu 与目前使用较广泛的检测引擎 Libemu 在不同行为类型的 Shellcode 检测方面的性能比较。

表 3 ExLibemu 和 Libemu 的检测能力比较

检测引擎 \ 性能指标	Call_jmp Getpc code	Fnstenv Getpc code	SHE Getpc code	Kernel32. dll base address resolution
Libemu	√	*	×	×
ExLibemu	√	√	√	√

由表 3 可知,ExLibemu 能正确检测上述行为类型的 Shellcode;而 Libemu 仅能正确检测 Call_jmp getpc code 行为类型的 Shellcode 以及部分 FnstenvGetpc code 行为类型的 Shellcode,无法检测 SHE getpc code、Kernel32. dll base address resolution 等行为类型的 Shellcode。因此,ExLibemu 的性能比 Libemu 的性能更好。

2)Web Shellcode 漏检率

在对 10000 个 Web 脚本样本进行检测时,WebMCDS 系统能正确检测出 1368 个不同类型的 Shellcode 样本,即检测引擎 ExLibemu 均能检测 FnstenvGetpc code、SHE getpc code、Kernel32. dll base address resolution 等行为类型的 Shellcode。对这些类型的 Shellcode 的检测正确率为 100%,漏检率为 0。具体如表 4 所列。

表 4 WebMCDS 检测性能

类型	数量	正确检测	误报	漏报
正常类型	8632	8497	135	0
FnstenvGepc code	546	546	0	0
SHE getpc code	203	203	0	0
Call_jmp Getpc code	292	292	0	0
Kere132. dll based address resolution	327	327	0	0

由表 4 可知,由于 WebMCDS 在检测过程中将 135 个正常页面误报为含 Shellcode 的页面,因此误报率为 1.35%。

结束语 Web 恶意代码通过客户端运行,能够入侵并窃取内网数据,引起严重的安全问题。如何能有效检测 Web 恶意代码仍是研究者共同关注的安全问题。文中重点研究跨站脚本漏洞、ActiveX 控件漏洞、Shellcode 等方面的检测,提出一种基于行为语义分析的 Web 恶意代码检测机制。首先爬取页面中的静态元素(关键的闭合标签),并进行语义分析;然后进行模拟执行,提取代码执行路径序列,进行分析和检测。实验结果分析表明,所提出的检测机制能准确地检测多种类型的 Web 恶意代码,而且漏检率低。

但浏览器及其插件还存在若干未知的 0day 漏洞,若这些 0day 漏洞被 Web 恶意代码利用,将会增加检测的难度。这种情景将是本文下一步的重点研究内容。

参 考 文 献

[1] Gu Xiao-dan, Yang Ming, Luo Jun-zhou, et al. Website Fingerprinting Attack Based on Hyperlink Relations[J]. Chinese Journal of Computers, 2015, 38(4): 831-845(in Chinese)
顾晓丹,杨明,罗军舟,等. 针对 SSH 匿名流量的网站指纹攻击方法[J]. 计算机学报,2015,38(4):831-845

[2] Kolbitsch C, Livshits B, Zorn B, et al. Rozzle: De-cloaking internet malware[C]// 2012 IEEE Symposium on Security and Privacy (SP). IEEE, 2012: 443-457

[3] Labuschagne W A, Eloff M M. Towards an automated security awareness system in a virtualized environment[C]// European Conference on Information Warfare and Security. 2012: 163-171

[4] Lu G, Chadha K, Debray S. A simple client-side defense against environment-dependent web-based malware[C]// 2013 8th International Conference on Malicious and Unwanted Software: "The Americas"(MALWARE). IEEE, 2013: 124-131

[5] Borgolte K, Kruegel C, Vigna G. Delta: automatic identification of unknown Web-based infection campaigns[C]// Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM, 2013: 109-120

[6] Chang J, Venkatasubramanian K K, West A G, et al. Analyzing and defending against web-based malware[J]. ACM Computing Surveys (CSUR), 2013, 45(4): 1-35

[7] Gu B, Zhang W, Bai X, et al. JSGuard: Shellcode Detection in JavaScript[M]// Security and Privacy in Communication Networks. Springer Berlin Heidelberg, 2013: 112-130

[8] Khodaverdi J. Enhancing the Effectiveness of Shellcode Detection by New Run-time Heuristics[J]. International Journal of Computer Science, 2013, 3(2): 2-11

[9] Zhang Hui-lin, Zou Wei, Han Xin-hui. Drive-by-Download Mechanisms and defenses[J]. Journal of Software, 2013, 24(4): 843-858(in Chinese)
张慧琳,邹维,韩心慧. 网页木马机理与防御技术[J]. 软件学报,2013,24(4):843-858

[10] Invernizzi L, Comparetti P M, Benvenuti S, et al. Evilseed: A guided approach to finding malicious web pages[C]// 2012 IEEE Symposium on Security and Privacy (SP). IEEE, 2012: 428-442

[11] Kapravelos A, Shoshitaishvili Y, Cova M, et al. Revolver: An Automated Approach to the Detection of Evasive Web-based Malware[C]// USENIX Security. 2013: 637-652

[12] Nazario J. PhoneyC: A virtual client honeypot[C]// Proc. of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET). Berkeley: USENIX Association, 2009