浏览器扩展安全缺陷检测工具的设计与实现

The Design and Implementation of Browser Extension Vulnerability Detecting Tool

学科专业：计算机应用技术
研 究 生：王　钦
指导教师：李晓红 教授

天津大学计算机科学与技术学院
二零一二年十一月

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **天津大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 **天津大学** 有关保留、使用学位论文的规定。特授权 **天津大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘 要

浏览器扩展可以为浏览器带来新的特性，而浏览器扩展机制允许用户利用第三方扩展为浏览器添加新的个性化功能，从而提升浏览器的性能和改变浏览器外观。然而，扩展机制的引入也同时为浏览器带来更多的安全相关的缺陷。现存一些针对扩展所引发的安全问题的研究并不能消除扩展安全缺陷为浏览器带来的威胁，而当扩展的安全缺陷需要通过其行为序列加以判断时，其引发的安全问题尤为突出。

在对火狐浏览器扩展的行为的大规模调查研究之后，本文以火狐浏览器为例，提出了一种基于行为序列安全性的浏览器扩展安全缺陷检测方法。方法中，首先需要插桩火狐原生浏览器，用于动态监测火狐扩展的动作，提取扩展运行过程中的所有相关的 XPCOM（跨平台组件对象模型）接口调用信息，并给出扩展安全初步检测结果以及扩展动作报告；同时需要完成扩展行为序列的研究，以及存在安全缺陷的行为序列的抽象和缺陷有限状态自动机的定义。继而，在监测平台和缺陷定义的基础上，本文给出扩展行为序列的化简方法，以及扩展缺陷自动判定方式和人工审查方式。最后，本文利用浏览器扩展安全缺陷检测方法对扩展进行实验，分析和验证本方法的有效性和效率。

通过实验和分析，本文中给出的浏览器扩展安全缺陷检测方法的有效性得到了一定程度的验证，并成功分析和判定出多类通过其它自动分析很难发现的安全缺陷，并找到了今后改进和研究的方向，即在效率上和缺陷定义的严格性和准确性上研究和改进。

关键词： 浏览器扩展 行为监测 动作序列 安全缺陷

ABSTRACT

Browser extension is a mechanism used to improve the performance and applicability of browsers by adding new personalization features, and it greatly facilitates users to modify their browsers to a variety of styles. However, extension mechanism may also introduce more security related vulnerabilities. Though the threats of malicious or vulnerable extensions have been addressed by several solutions, these solutions are still far from eliminating the threats, especially when the extensions' behavior sequences are unsafe.

After a large-scale study to the behaviors of Firefox browser extensions, using Firefox as an example, this paper provided an approach firstly to synthesize abstract behavioral models from XPCOM（Cross Platform Component Object Model）interfaces, invoking sequences of extensions obtained by the run-time interface invoking approach, which requires the preparatory implementation of a behavior monitoring system. Secondly, it purposes to define the vulnerable behavior sequence patterns that are used to guide the testing process adopted on sequence matching methods for detecting the security and reliability vulnerability of extensions. The effectiveness and performance of our approach and detecting tool is provided with the support of experimental results.

Through the experiment and the analysis on it, the effectiveness of the approach provided has been verified, and although, the further research and improving direction is found. The efficiency improvement and more sophisticated definitions on vulnerable behavior sequence are needed in the future.

KEY WORDS： Browser extension, behavior monitoring, behavior sequence, vulnerability

目　　录

第一章 绪论

1.1 研究背景及意义

随着网络技术的迅猛发展，各型各类网络应用极大的丰富了人们的工作和生活。浏览器，作为人们访问这些网络应用的窗口，如今已经毋庸置疑地成为了人们使用频率最高的一个应用。作为绝大多数计算机与网络交互的窗口，浏览器极大地方便了人们电子商务、个人娱乐、社交网络等多种在线服务。浏览器应用的广泛性以及重要性使得它的安全一直以来都是安全领域研究人员关注的重点，并且随着浏览器支持的 web 标准的演进（由最初仅用于处理和渲染静态页面的应用软件演变为如今可以支持如 XML、SVG、JavaScript、Ajax 等众多 Web 标准），以及允许第三方为浏览器添加新的功能特性的浏览器扩展机制[1]的引入，使得浏览器的安全问题更为突出。同时，主流的浏览器（如 IE、Firefox 以及 Chrome）都支持扩展机制，绝大多数用户可以选择安装合适的扩展来为浏览器添加新的功能以满足其个性化的功能需求。

然而，功能不断强化、多样化的浏览器，在广泛的、高频度的使用下，逐渐成为黑客们的攻击目标，加之浏览器自身设计和实现以及其扩展和插件机制存在的诸多安全方面的缺陷，使得其安全性问题不容忽视。据赛门铁克[1]统计，从 2006 年 8 月份到 12 月，火狐浏览器发现了 40 个安全漏洞，在此期间，微软 IE 发现了 54 个漏洞，挪威 Opera 和苹果的 Safari 浏览器各自有四个漏洞。而从 2006 年全年统计来看，火狐和 IE 的漏洞“旗鼓相当”，火狐发现了 87 个漏洞，IE 则是 92 个。赛门铁克公司的统计还显示，在修补安全漏洞的速度上，Mozilla 比 Microsoft 快五倍。火狐从漏洞发现到修补漏洞的时间平均为两天，而微软是十天。同时其安全响应团队负责人 Weafer 表示，浏览器是一个安全漏洞集中的地方。在金山公司发布的《金山安全 2010 木马发展趋势报告》中，最容易被绑架型木马利用的软件前两位分别是浏览器和输入法，并且浏览器相关组件排在十种木马最喜欢攻击的系统应用第二位。

安全管理公司 Qualys[5]的最新一项研究显示，绝大多数网络浏览器上运行有过时插件，导致浏览器在安全攻击面前变得不堪一击。通过对浏览器检查工具的 42 万份扫描样本进行分析，Qualys 发现，最大的问题都出在几个常见视频插件——如 Adobe Flash[4]、苹果 Quicktime[5]、Shockwave 以及 Windows Media Player，PDF Reader 阅读器这类更加通用的工具，以及 Java 虚拟机[6]上。安全性最差的

插件是 Java。Java 在 80%的浏览器上都有安装，其中 40%的浏览器运行着极易遭受攻击的老版本 Java。Adobe Reader 阅读器的安全性排名第二，其在浏览器上的安装率也是 80%，其中逾 30%的浏览器易遭受攻击。而更常见的 Flash 视频插件也是个问题，有 95%以上的浏览器安装有 Flash 视频插件，而其中 20%的浏览器易遭受攻击。其余的视频播放器 如 Shockwave 和 Quicktime 易受攻击的程度在 20%—25%之间，不过只有大约 40%的浏览器安装了这两个插件。总体来说，如今与浏览器(无论是哪种浏览器)相关的安全漏洞大约有 80%是插件导致的，而只有 20%的漏洞是浏览器的。说明浏览器的扩展机制在为用户带来良好体验的同时，也为用户使用浏览器增添了更多的不安全因素。近些年人们在浏览器扩展中发现的大量漏洞[7]，一方面给用户不加选择的使用浏览器扩展敲响了警钟，另一方面也使更多的研究人员关注于如何解决浏览器扩展的安全问题。

对于浏览器安全问题的研究，近些年来研究人员取得了一定成果，同时也发现了扩展机制带来安全漏洞的根源，即扩展机制赋予扩展太高的权限，但是却并没有限制和保障权限不被滥用的完善的安全机制[7]。在限制浏览器中扩展的高权限问题上，研究人员给出了一些解决方案，如分散、隔离等；也给出了一些可以缓和扩展安全问题的方法，如静态分析、动态监测及浏览器扩展机制新设计方案等。但是，却很少有工作重点关注当前浏览器扩展行为的分析、检测与调查研究，而对扩展的行为内容和动作序列的关注更为不足，使得至今人们对于浏览器扩展市场上扩展的安全现状也没有一个清晰的认识，并且没有一套高效精确的浏览器扩展安全缺陷检测方法。因而，本文中提出的针对扩展的行为安全性和动作序列安全性的浏览器扩展安全缺陷监测方法，可以较大程度上为浏览器解决扩展带来的安全问题，具有重要意义。

1.2 研究内容

综上所述，虽然目前浏览器扩展的安全问题已经受到了相关研究人员的关注，但很少有人对浏览器扩展的行为进行详细的分析，并对浏览器的行为序列安全性进行调查研究和缺陷模型抽象，针对性的设计和实现浏览器扩展安全缺陷检测方法。而这些正是本文要研究的主要内容，具体包括：

1. 浏览器扩展行为的分析、抽象和扩展缺陷建模。通过对浏览器为扩展提供的接口进行分析，本文中主要是研究 Firefox 对扩展提供的 XPCOM 接口，以对浏览器扩展的行为进行分类，并对存在缺陷的扩展行为序列抽象和建模。

2. 浏览器扩展行为监测平台的设计和实现。实现整个监测系统，完成动态的扩展接口调用抽取功能，实现基于浏览器扩展行为分析的扩展动作报告生成和扩展行为序列的化简处理模块。

3. 基于行为序列的安全缺陷检测方法的研究以及检测工具的设计和实现。基于浏览器扩展行为监测系统的支持，分析扩展对接口的实时调用信息，研究扩展行为，设计应用序列匹配的安全缺陷检测方法及检测工具。

4. 实验设计和实施。对现存的存在安全缺陷和不存在安全缺陷的扩展进行有效性实验。

1.3 论文结构

文章的主体分为如下几个部分：绪论；浏览器扩展的安全；浏览器扩展安全缺陷检测方法研究；浏览器扩展行为序列检测工具实现；实验分析；总结与展望；参考文献；发表论文和参加科研情况说明；致谢。

本文旨在利用自行设计的浏览器扩展安全缺陷检测方法，对浏览器扩展的行为信息以及动作序列的监测和提取，继而完成动作序列检测方法设计和检测工具实现；应用设计的检测方法进行实验，并对实验结果进行分析研究以期获得对后续检测方法研究和改进工作具有指导意义的信息。文章主要章节的编排如下：

第一章 主要阐述课题的背景和意义，提出本文的研究内容，并对文章的结构做出安排。

第二章 对浏览器扩展安全相关的国内外研究工作进行综述，包括浏览器扩展安全概述、浏览器扩展行为调查以及浏览器扩展的检测方法。

第三章 对浏览器扩展安全缺陷检测方法和检测工具的设计进行详细的阐述，包括检测方法的阐述和扩展行为监测系统的架构以及监测流程，以及对扩展的监测信息的处理和行为序列安全性判定方法。

第四章 详细阐述了浏览器扩展安全检测工具的实现。对系统实现中用到的工具进行了详细介绍，并对系统各个模块的实现以及在实现过程中遇到的挑战及解决方法进行了细致的阐述。

第五章 在对安全和存在安全缺陷的浏览器扩展进行测试之上，对实验结果进行了详细的分析，对检测方法的有效性和效率进行分析，并对实验方法的改进进行了探讨。

第六章 结论与展望，总结了本文的工作及项目中存在的问题并提出了未来的研究方向。

第二章 研究综述

浏览器扩展机制允许用户应用第三方开发的扩展为浏览器添加个性化功能，极大的加强了浏览器的表现力和自由度。但同时，扩展机制也将浏览器暴露于更多的威胁之下。本章重点介绍国内外近年对浏览器扩展安全问题研究状况，其中包括浏览器扩展安全概述，浏览器扩展行为研究和分类，以及动态和静态的浏览器检测方法三部分。这些研究工作为本文的方法研究提供了理论和实践基础。

2.1 浏览器扩展安全概述

随着网络技术的迅猛发展，各型各类网络应用极大的丰富了人们的工作和生活。浏览器，作为人们访问这些网络应用的窗口，如今已经毋庸置疑地成为了人们使用频率最高的一个应用。主流的浏览器（如 IE、Firefox 以及 Chrome）都支持扩展机制。浏览器的扩展机制允许第三方为浏览器添加新的功能特性。用户可以选择安装合适的扩展来为浏览器添加新的功能以满足其个性化的功能需求。

关于浏览器扩展，Mozilla 的官方文档[8]给出了如下定义：“扩展为 Mozilla 的应用程序（如 Firefox、SeaMonkey）添加了新的功能，添加的功能可以是一个工具栏按钮也可以是一个全新的功能特性。它允许用户可以通过定制应用程序来满足其自身的个性化需求。”

为满足多种功能方面的需求，扩展往往会申请并获得很高的权限。在高权限下，扩展可以监测、改变网页内容，访问存储的某些网站的用户名和密码以及一些 cookie 信息，以用户之名发送 http 请求，或者建立 socket 以监听或者连接远程服务器，甚至于访问本地文件系统和开启本地进程，这使得扩展几乎具有了与浏览器相同的权限。这些特权的提供，使得扩展可以更好的扩充浏览器功能，但是也会给浏览器带来更多的攻击。

Liverani[7]对已经发现的恶意或有安全漏洞的扩展（如 Coolpreview2.7、skype3.8、UpdateScanner3.0 等）进行了总结，并详细分析了其原因及攻击方法。文献[9]中给出已识别出的五种常见的攻击场景：创建按键监听器，损坏访问的网页，钓鱼攻击，盗取密码以及将 Firefox 作为僵尸。而在这些攻击场景之下，通过扩展攻击 Firefox 的方法可以分为三种：跨站脚本攻击（XSS）、安装恶意的扩展[36]以及修改已安装的扩展。安全评估组织 Security-Assessment.com 发布的白皮书[10][11]中，对常见的攻击 Firefox 扩展的恶意代码进行了总结，如访问本地文

件、远程代码执行以及盗取密码等，并详细总结了八类可能被攻击者通过扩展进行 XCS（Cross Context Scripting）攻击的缺陷，分别是拖拽的事件处理器 XCS 攻击、自定义 DOM 事件处理器攻击、跨域内容/脚本、XBL（XML 绑定语言）[35]注入、封装器（Wrapper）攻击、XPCOM 攻击、泄露沙箱对象、绕过 nsIScriptableUnescapeHTML.parseFragment()方法。

Firefox 的扩展系统[1][13][28]如图 1 所示。XPInstall[29]模块主要负责扩展的安装，它将一个具有特定格式的扩展压缩文件安装到 Firefox 的目录中。一旦扩展被安装到 Firefox 中，将由 XULRunner[12][30]平台负责扩展的运行，该平台同时也是众多 Mozilla 应用（如 Firefox、Thunderbird）的运行平台。在 XULRunner 平台中，具体由 SpiderMonkey[40] JavaScript 引擎负责扩展中 JavaScript 的解释执行，XPConnect 模块作为 SpiderMonkey 与 XPCOM 组件的桥梁，使得扩展可以访问诸多底层的 XPCOM 组件，如负责页面的布局和渲染的 Gecko 组件、负责网络访问的 Necko 组件以及负责访问本地文件系统的组件等。XPCOM 技术[32][33]是 Mozilla 技术核心，它提供了一个框架允许开发人员将大型软件项目分解为多个小的、可重用的组件对象，然后可以在运行时动态地进行装配以实现功能的整合。

在 Firefox 的扩展系统中，扩展被赋予了极高的运行权限。它不仅可以访问用户浏览的页面内容，还可以访问网络以及本地文件系统，甚至可以开启一个本地进程。总之，Firefox 中的扩展几乎具有与 Firefox 自身同等的权限。Firefox 的这种扩展机制的设计一方面使得扩展能够极大的丰富浏览器的功能，但同时也为浏览器的安全问题带来了隐患。

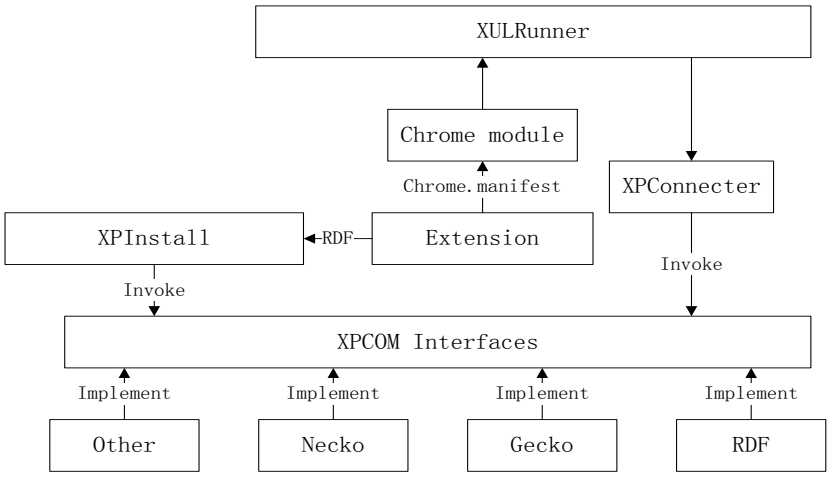


图 2-1 Firefox 扩展系统结构图

XULRunner 平台负责扩展的运行，而扩展通过 XULRunner 可以访问诸多底层的 XPCOM 接口，从而实现各种所需功能，同时 XPCOM 接口的访问，使得扩

展可以获取高权限。Mozilla 采取了多种措施[14]来缓和 Firefox 的扩展安全问题，具体包括：

（1）签名机制

为了确保扩展确实来源于其声称的第三方开发者，以及扩展在传输过程中没有遭到恶意篡改，Mozilla 提供了签名机制，使得开发者可以对扩展进行签名，用户在安装扩展时对签名进行校验。然而，由于签名并不是强制性的，因而扩展的开发者并不愿花费太多的成本来对扩展进行签名，并且签名机制在确保扩展安全性方面作用有限，导致绝大部分 Mozilla 发布的扩展都没有签名。

（2）审查机制

为了发现并阻止开发者提交恶意或有安全漏洞的扩展，Mozilla 对新提交的扩展在发布之前要进行一系列审查，对于未通过的扩展不予发布，从而一定程度降低了恶意扩展流入最终用户的可能性。然而，Mozilla 对于扩展的审查基本上是在简单工具的帮助下人工对扩展的代码进行审查。这就使得审查过程伴随有很多的人为因素，从而无法对扩展的安全性提供足够的保证。

（3）JavaScript 沙箱

JavaScript 沙箱机制使得扩展与网页之间没有直接的信息交互。沙箱系统采用 XPCNativeWrapper 对象[15]支持扩展间接地与网页进行通讯，它将来自于不可信域的对象包裹起来，使得拥有特权的代码可以以安全的方式访问该对象。它通过限制对被包裹对象的属性和方法的访问，阻止攻击者在网页中对扩展的属性和方法进行重定义。虽然沙箱机制可以在一定程度上避免扩展中安全漏洞的产生，但是由于它可以被扩展的开发者所关闭，因而也不足以保证扩展使用的安全性。

Firefox 的安全机制并不能很好的限制扩展权限和解决扩展的安全问题，Sruthi Bandhakavi 等人基于静态信息流分析方法[16]，对扩展中 Javascript 代码进行静态分析，并构建了 VEX[17]，用于检测 Firefox 扩展中的安全缺陷，并且发现了数以百计的不良扩展开发实践所导致的安全缺陷。但其中工作主要是手工完成，工作量大，且需专人分析。为限制扩展权限的滥用，Adam Barth 提出针对谷歌浏览器的最小权限方法[18]，重新设计扩展系统的结构，控制扩展的权限使用。文献[19][20]中，作者提出了一种扩展隔离机制来加强浏览器的安全，即通过保护浏览器代码库的完整性和用户信息的保密性以及完整性的方法。虽然这些方法同 Firefox 的扩展安全机制一起在某种程度上保护了浏览器的安全，但是依然不能很好的解决扩恶意展和存在安全缺陷的扩展所带来的安全威胁。

2.2 浏览器扩展行为研究与分类

Firefox 浏览器向扩展暴露的接口： ①JavaScript 函数；②XPCOM 接口。从安全的角度考虑，XPCOM 接口由于其本身具有强大的功能，因而隐藏着巨大的安全风险，是本文重点研究的对象。

在 Firefox 版本的演进过程中，XPCOM 接口也会随之进行一定程度的调整，本文的研究工作主要是针对 Firefox 3.5 进行的。本文对 Firefox 3.5 中的 1448 个 XPCOM 接口进行了研究，按照接口提供的功能，将其划分为 57 种不同的行为类别，如附录 A 所示。这 57 个行为类中，大部分属于扩展或浏览器正常的功能需求。然而，有些行为存在一定的安全隐患，如对密码管理器进行访问的“Password”等，这些行为可能属于浏览器扩展的正常需求，但也存在恶意扩展利用这些接口窃取用户敏感信息的可能。根据各个接口行为类别与安全的相关程度，同时将 57 个接口行为类别划分成 4 个安全相关度递减的组，见表 2-1。

表 2-1 接口行为安全相关度分组

安全相关度	扩展行为分组
紧密	Process launching, Thread management, Clipboard access(belong to Accessibility), Arbitrary file access, Download, XPInstall, Network access via XPCOMAPI(belong to Network access), Update, DOM injection(belong to Doc handling)
一般	Password, Login, Cookie, Preference, Profile, Network access(via XMLHTTPRequest), Addons management
松散	History, Bookmark, Dynamic code execution
无关	Accessibility, Browser core, Auto complete, Log to console, Searching, Spell checking, DOM, Editor, Internationalization, Offline cache, XML parser, Network utilities, Parser, Channels, Protocol handlers, Sockets, Proxies, RSS/RDF, Data types and structures, Streams, Memory management, Component management, Additional XPCOM services, Javascript core, Javascript debugger, XPConnect, Authentication, Certification, Crptograph, Additional security interfaces, Doc handling, Transaction management, Web worker, Window management, Print, Other tools, Storage, Images, ZIP/JAR process, JVM, Plugins

总之，通过对 Firefox 向扩展暴露的所有 XPCOM 接口以及部分安全相关的 JavaScript 函数的分析，基于其在功能上的相关性，本文定义了 57 种 Firefox 浏览器扩展的行为。并利用这一行为分类的定义，指导行为监测过程中对扩展安全性的判断，以及接口调用序列的约简处理方法。

2.3 浏览器扩展检测方法

除了对浏览器扩展的安全问题的原因、本质进行深入研究以及对浏览器扩展的安全现状进行调查之外，相关研究人员同时也在不断探索如何对恶意的或有安全漏洞的扩展进行检测。关于如何有效地检测并缓和浏览器扩展中的安全缺陷，目前研究人员已经取得了一定的成果。具体方法主要包括如下两大类：

2.3.1 静态检测

静态检测[21-22]的主要思想是将待检测的内容做代码级分析，并将安全缺陷定义为特定的模式，根据定义的模式查询代码分析中的是否存在该模式也即该类型的缺陷。其中的技术关键点和难点都在于缺陷模式的构建，局限性在于只能检测已知类型的缺陷和预定义的缺陷类型，查全率低。

文献[21]提出了一种 GateKeeper 静态分析方法，该方法首先要求提交的扩展必须满足 JavaScriptSAFE（分析 JavaScript 语言的静态子集）语法规范，在其范围内分析 JavaScript 代码，并利用 DataLog 逻辑系统演绎系统进行上下文敏感的指向分析；而对于不满足 JavaScriptSAFE 的代码则不能分析。在分析过程中的最后阶段，需要在指向分析的结果之上分析其是否满足预定义的安全及可靠性策略，如限制扩展的权限、检测是否重写了封存对象（Frozen Objects）、检测代码注入、重定向浏览器以及其他一些与主机相关的策略。

文献[22]则提出一个称为 VEX 的基于信息流分析的浏览器扩展安全缺陷检测框架，该框架对扩展的 JavaScript 代码进行上下文敏感、流敏感的静态分析，通过遍历 JavaScript 的 AST 树（抽象语法树）来计算出所有从敏感的（需要关注的）的源（Source）到敏感的信息宿（Sink）的信息流，并在此基础上识别预定义的缺陷模式及不安全的编程实践。文中关注的安全缺陷模式及不安全编程实践主要有：网页内容到 eval 模式、网页内容到 innerHTML 模式、RDF 到 innerHTML 模式、evalInSandbox 对象到“==”或“！=”模式以及对 WrappedJSObject 对象的调用模式。但是，该方法对于 JavaScript 语言的许多特性都不能很好的支持（如 Prototype、eval），导致其实用性及检测效果并不理想。此外，该方法所基于的信息流模式归纳的并不完全，其所覆盖的漏洞范围不是很大，导致不可能达到很高的查全率。

上述方法最大的不足在于，无法对扩展在运行时动态生成的代码进行检测（如通过 Eval 函数将输入字符串转换成的代码），而这种特性在浏览器扩展中往往又是非常常见的，且很容易产生安全缺陷。同时，该方法只适用于对已知缺陷的检测，很难通过该方法发现从未出现过的安全缺陷，以及间接缺陷。

2.3.2 动态监控

动态监控[23-27]是指在程序运行的过程中，动态分析其行为，并根据预定义策略对可能的恶意行为加以识别的一类方法。

Sabre[23]通过监控浏览器内部的信息流来对浏览器扩展进行分析，当发现扩展以不安全的方式访问敏感信息（如将 cookie 信息发往不可信站点）时，系统将产生警告。然而，动态信息流分析仅对于那些直接通过浏览器盗取敏感信息的攻击具有较好的检测效果，对于那些利用浏览器扩展漏洞下载木马，然后再利用木马盗取敏感信息的攻击，则没有任何防御效果。

文献[24][25]讨论了浏览器扩展行为的运行时监控技术，提出通过控制扩展对 XPCOM 的访问来降低扩展的安全风险。然而，扩展的安全风险不仅仅是因为其对 XPCOM 的访问，对某些 JavaScript 引擎内部函数的访问（如 Eval、Function）也会造成很高的安全风险，因而，仅仅对 XPCOM 的访问进行访问控制对于浏览器扩展安全问题的解决显然是不完全的。

文献[26]]设计了一个用于监控 JavaScript 代码执行的审计系统，该系统可以在浏览器扩展运行的过程中对扩展调用的 JavaScript 函数、访问的 XPCOM 接口甚至是为其传递参数进行记录。文中还提出了基于该系统进行恶意代码检测的方法，并给出了针对跨站脚本攻击、恶意的 window.open 以及 window.alert 调用的检测案例。但是该系统缺乏实时性。

文献[27]则是针对 Internet Explorer 设计了一种扩展行为的控制机制，它通过在扩展与浏览器之间插入一个代理模块，来控制它们之间的交互。当扩展请求访问浏览器核心组件时，代理模块会按照预定义的安全策略来对这一请求进行审计。如果这一请求没有违反任何预定义的安全策略，则请求将会被转发给浏览器核心组件以完成扩展与浏览器核心的一次交互。否则，这一请求将被代理模块否决。类似地，浏览器核心到扩展之间的信息传递也需要经历这样一个过程。

上述方法虽然可以有效地检测出恶意或有安全漏洞的浏览器扩展，但其存在的缺点也是显而易见的：一方面由于动态检测很难覆盖扩展的所有控制流，因而检测结果中必然存在漏报的情况；另一方面由于动态检测一般都需要在浏览器原有代码上添加新的模块或代码以监测扩展的运行，因而必然会为浏览器的正常运行引入很大的负载。

本文主要工作是设计浏览器扩展缺陷检测方法，主要是通过自行设计并实现的浏览器扩展行为监测系统对浏览器扩展的行为进行实时的动态的监测，并提取扩展运行时对底层 XPCOM 接口调用信息，异步的分析所截获的信息中包含的不安全因素，以期更好地解决浏览器扩展的安全缺陷检测问题，属于动态分析和检

测范畴。但是与现有工作的不同之处在于，本文通过自行开发的浏览器扩展行为监测系统对浏览器扩展进行运行时的实时监测，并在监测结果基础上对浏览器扩展的行为序列进行异步的缺陷检测，因而产生的数据以及分析结果更具真实性及说服力。同时，本文中提出对浏览器扩展的行为也即动作序列进行检测的方法是其他的工作没有涉及的。

第三章 浏览器扩展安全缺陷检测工具设计

为了有效降低恶意或存在安全漏洞的浏览器扩展为用户带来的安全风险，本文在对 Firefox3.5 的 XPCOM 接口进行了详细研究的基础上，对扩展的行为（所调用的接口的行为）进行了分类、定义，设计并实现了一个针对 Firefox 浏览器的扩展行为监测系统，将其作为扩展行为监测方法中的扩展运行时动作监测平台。在行为监测的输出结果之上，利用接口序列匹配方法确定扩展行为序列的安全性。

在本文设计的浏览器扩展安全缺陷检测过程中，首先需要将欲检测的扩展下载到本地扩展库，以其为扩展监测系统（即插桩的浏览器）提供输入；扩展监测系统在运行时抽取扩展对 XPCOM 接口的实时调用信息，并对接口序列进行化简处理，为序列匹配工具提供行为序列的输入，并为人工审查提供扩展在运行时的动作报告；序列匹配工具对扩展行为序列进行检测，给出匹配和检测结果；通过对扩展动作报告和扩展动作序列匹配结果的综合分析，给出最终的扩展安全性检测结果。整个浏览器扩展安全缺陷检测过程如图 3-1 所示：

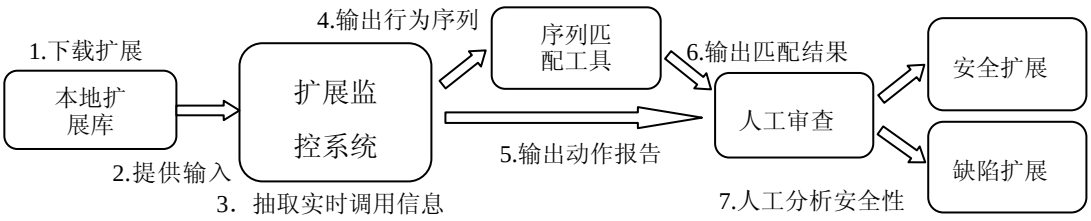


图 3-1 缺陷检测过程

3.1 浏览器扩展行为监测

本文中设计的扩展行为监测方法中，需要得到实时的、扩展对 XPCOM 接口调用信息，并根据接口调用信息初步判断扩展安全性和进一步的接口调用信息的处理。而方法的实现，需要借助插桩浏览器的方法实现，也即通过插桩原生的浏览器，构建浏览器扩展行为监测系统，以实现和验证方法。该方法中，如果根据单个接口调用信息可以确定当前监测扩展存在安全缺陷，则报告缺陷，然后继续给出扩展动作报告；如果监测系统不能确定扩展存在安全缺陷，则将记录的扩展对接口的调用信息进行约简处理（由监测平台中的接口序列约简部件完成），并

生成扩展动作报告，便于扩展安全性的人工审查，同时需要给出约简后的扩展动作序列，用于进一步的扩展动作序列安全性监测。

3.1.1 浏览器扩展行为监测系统框架

扩展行为监测系统框架如图 3-2 所示：

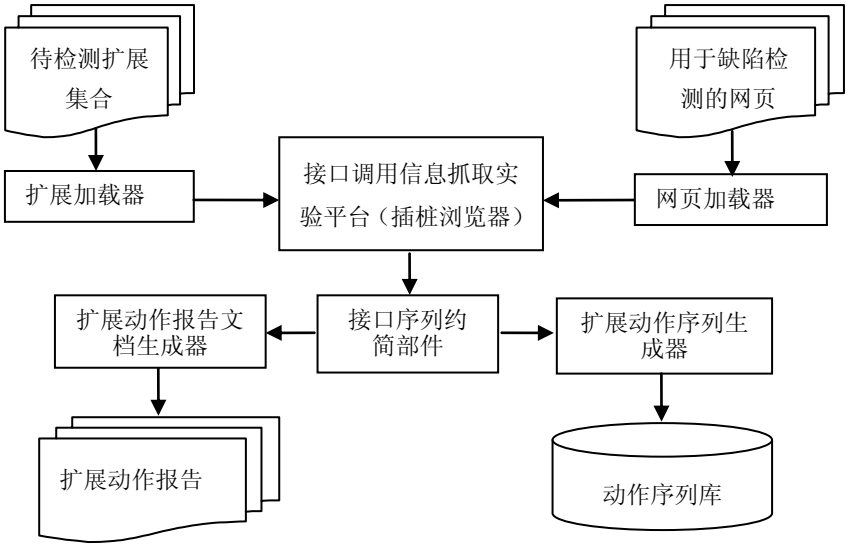


图 3-2 行为监测系统架构图

图 3-2 中，待检测扩展集合为本地扩展库，用于存储下载自 Mozilla 官网的待检测扩展，辅助扩展加载器为插桩浏览器提供自动安装和卸载功能；扩展加载器为接口调用信息抓取平台提供待检测扩展的自动安装和卸载功能。

用于缺陷检测的网页是自主搭建的简单网站，用于接收和反馈插桩浏览器在访问该网址时的操作，如按钮点击，用户输入等，方便插桩浏览器对扩展运行时调用底层 XPCOM 接口信息，配合接口调用信息抓取。

插桩浏览器即为在浏览器源码中特定部分插入编写的具有监测和信息抓取功能的钩子代码重新编译的改动后的实验平台，用于实时监测扩展在访问指定网站并模拟用户操作时对 XPCOM 接口调用信息，其中包括调用时间（InTi），调用函数（InFa）和用户行为模拟的动作编号（AcNum），同时完成信息的记录功能。为了获得这些信息且达到较好的效率及性能，本文将桩模块设计为浏览器的一个动态链接库。如图 3-3 所示，库中包含了一系列设计良好的钩子，通过将其插入到浏览器源代码的适当位置来截获不同的浏览器内部事件，如对关键接口的访问、危险 JavaScript 函数的调用等等，可以应用其对扩展的安全性做出简单、初步的判断。

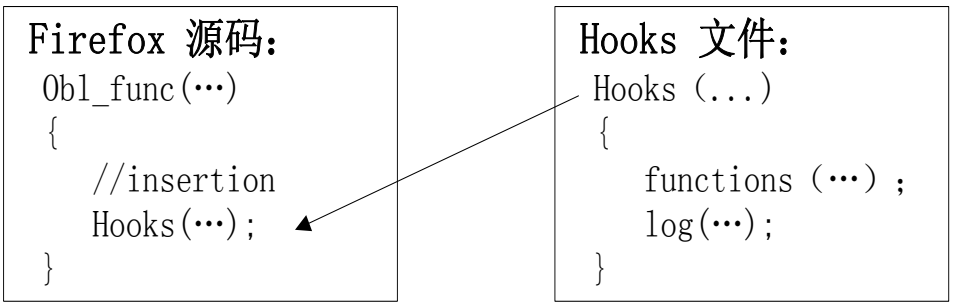


图 3-3 钩子示意图

接口序列约简模块负责对接口调用信息整理，包括对所有用户行为块中的接口调用序列进行全排列，完成接口序列中连续重复出现的接口名去重工作，实现接口序列约简功能，完成接口名到接口所属动作类的映射，以及对扩展行为序列化简。

扩展动作报告文档生成器用于接收接口序列约简模块的输出，并根据化简后的扩展动作序列等信息综合生成包括扩展的动作类别数目，动作类别的组合方式，以及动作序列的扩展动作报告。

扩展动作序列生成器用于接受接口序列约简模块的输出，并根据树形结构的动作序列输入，按照分支拆分出多条可用于动作序列匹配工具输入的迹。

扩展行为监测系统输出的扩展动作报告和扩展动作序列分别用于人工审查和动作序列检测工具的输入。

3.1.2 浏览器扩展行为监测系统监测过程

在浏览器扩展行为监测系统运行开始后，首先在插桩浏览器上安装指定的待检测扩展，并保证无其他扩展安装在插装浏览器上，以使得监测和记录的信息均为所检测扩展对底层接口的调用信息。在安装指定扩展之后，应用插桩浏览器访问构建的可以接受用户输入、点击等操作并做出反应的网站，并保证浏览器在每个用户操作之后进入睡眠状态一定时间间隔，用以通过时间间隔区分用户的单个操作下，扩展的所有 XPCOM 接口调用信息。

实验过程中，需要对扩展对接口的调用信息进行监测，监测过程中如果发现直接缺陷，即非接口调用序列构成的安全缺陷，则报告缺陷，缺陷报告后仍需要记录接口调用信息，用以生成扩展的动作报告；如不能发现直接缺陷，则记录扩展对接口的调用信息。

接口调用信息收集并记录之后，将接口调用信息记录传递给接口序列约简部件，并由该部件对接口调用信息按照约简步骤进行处理，即：首先从接口调用信息中抽取接口调用序列，继而通过去掉接口调用序列中连续重复出现的接口名（连续重复出现时，保留第一个，其余从序列中删除），并削减掉序列中出现的

非安全相关的接口名，从而化简接口调用序列；然后根据序列中各个接口名所对应的动作类别，完成接口调用序列到动作序列的映射；最后化简得到的动作序列。

根据获得约简后的动作序列，分别完成扩展动作报告的生成和动作序列检测工具的输入（扩展动作迹）的拆分，并记录和存储，完成扩展行为监测和接口调用信息处理过程。

浏览器扩展行为监测系统的监测流程图如图 3-4 所示。

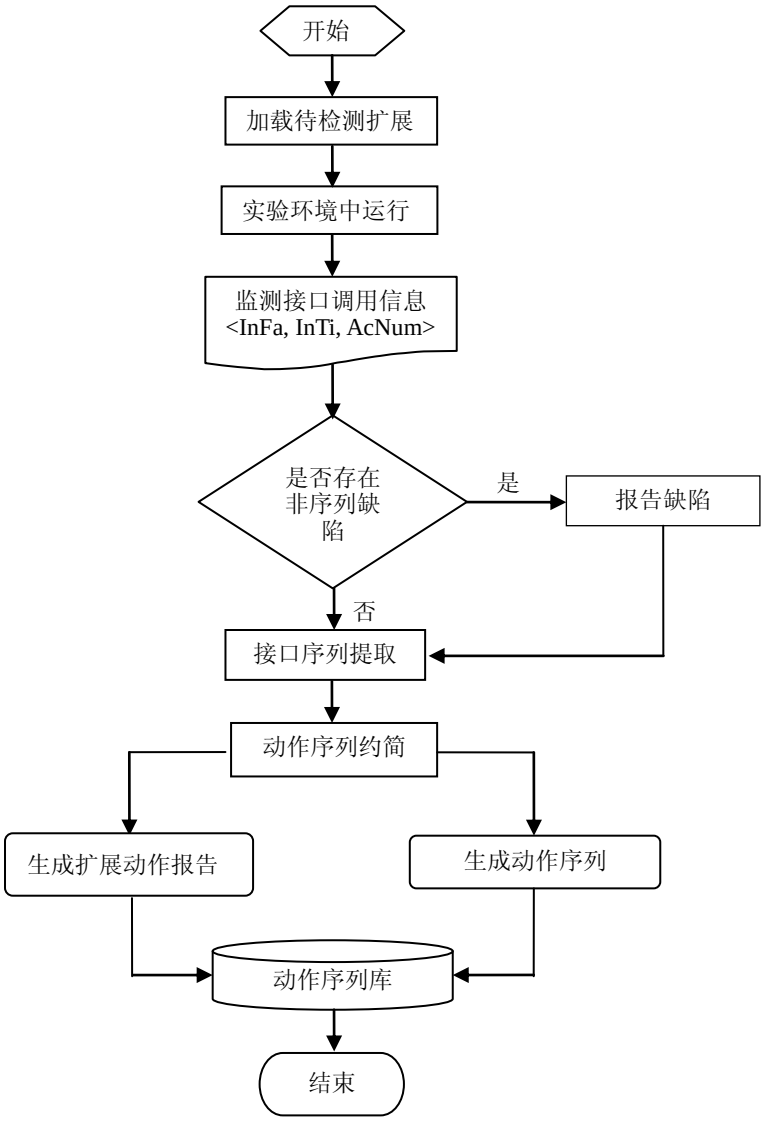


图 3-4 行为监测系统工作流程图

3.2 扩展接口调用序列处理

在整个扩展行为监测系统的运行过程中，扩展对 XPCOM 接口调用信息的收集和处理是重点内容之一。接口调用信息包括接口名，参数，调用函数，接口调

用时间，以及引发当前接口调用的用户行为号（用于区分每个用户行为引发的接口调用）等。根据接口调用信息，可以分析出扩展中的直接缺陷，如不能发现直接缺陷，需要对接口调用序列进行化简，继而得到扩展动作序列，并根据动作序列进一步判断扩展的安全性。

对接口调用信息的处理过程如图 3-5 所示。

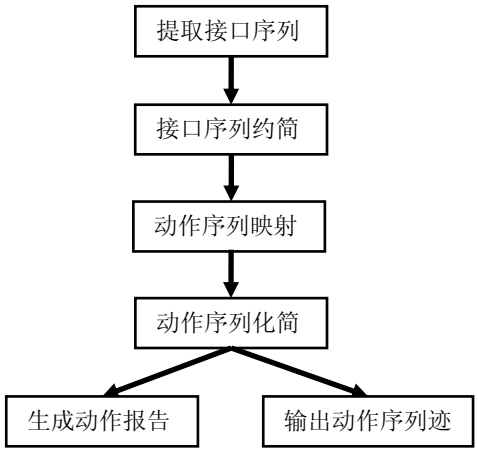


图 3-5 接口调用信息处理流程

3.2.3 提取接口序列

从扩展监测系统中的插桩浏览器收集到的扩展对 XPCOM 接口的调用信息中提取出扩展对接口掉调用序列，并根据不同的用户行为号分组，并对分组后（组内有序，时间序）的接口进行组间的全排列，以在简化实验过程基础上得到较为全面的信息。例如：键盘输入行为引发的扩展对接口调用和鼠标点击行为引发的扩展对接口的调用可以排列为两种形式，键盘输入在前，或者鼠标点击在前。

3.2.4 接口序列约简

在得到接口序列之后，将实验结果中连续重复出现的接口名进行去重处理，即保留连续重复出现的接口名的第一个，其余则从序列中删除。去掉连续重复的接口名之后的序列仍会比较复杂，且其中包含一些对扩展的安全性几乎没有影响的接口名（这累接口名也无需映射到行为序列），所以需要进一步处理，即删除序列中的这类接口名。

3.2.5 动作序列映射

接口序列到动作序列的映射。XPCOM 提供的接口有其所对应的功能，也即帮助浏览器扩展做出相应的动作，所以接口名序列需要映射到动作序列上。根据

序列中不同的接口名所属的动作类别，对其进行接口名到动作类别的替换。由于不同的接口名可能对应相同的动作类别，映射过程也是一个序列约简过程。

3.2.6 行为序列化简

由于不同接口名可能对应相同的动作类别，于是需要对替换结果做如 3.3.2 中所述的去掉重复项处理。并且因为相同的动作可能出现多次，从而使得行为序列中出现环，所以需要去掉序列中的环，并将动作序列处理成树状结构，从而完成行为序列化简。

3.2.7 生成动作报告

扩展动作报告可以作为扩展主要功能分析的重要参考信息，并可以用于辅助人工扩展安全性判断和审查。根据简化后的扩展动作序列结果，给出扩展主要动作统计，和扩展的主要功能分析，并将主要的扩展动作序列放入扩展的动作报告，完成报告生成。

3.2.8 输出动作序列迹

动作序列迹为无环无分支（无重复项）的线性序列。根据简化后的扩展行为序列结果，从其树形结构的序列上，按照不同分支，自顶分解出所有的分支行为序列，每条序列都需要包括树形序列的头结点动作，即都需要从头结点开始，以作为扩展动作序列检测工具的输入。扩展写为序列迹拆分方法如图 3-6 所示。

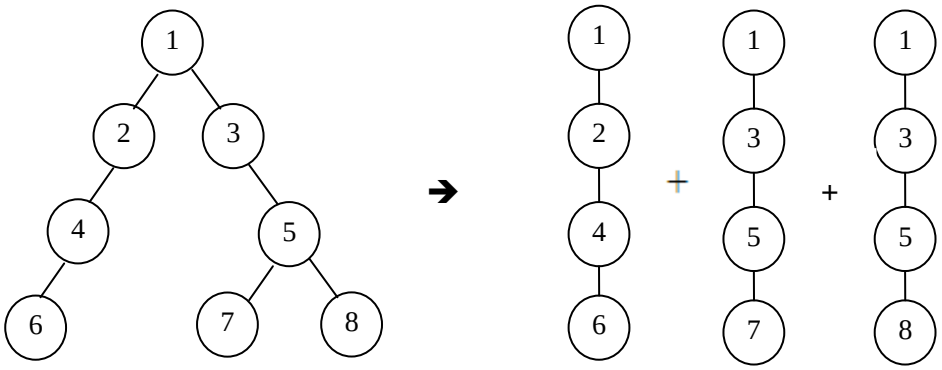


图 3-6 行为序列拆分方法示意图

3.3 浏览器扩展的动作序列检测

对于一些在行为监测系统中不能发现安全缺陷的扩展，并不能排除其存在安全缺陷的可能性，需要对其动作序列进行检测，从而判断其安全性。本文中利用缺陷行为序列定义，查找被检测扩展中的安全缺陷。在缺陷序列匹配过程中，查询待检测扩展的动作序列中是否存在缺陷动作序列的匹配，匹配结果包括完全匹配，部分匹配和不匹配，根据匹配结果，决定后续人工审查的审查内容。浏览器扩展动作序列迹的匹配结果是判断扩展安全性的依据。

3.3.1 扩展缺陷动作序列定义

Firefox 3.5 中有 XPCOM 接口 1448 个，可划分为 57 个动作类别，如附录 A 所示。根据动作分类的研究和分析，并针对存在安全缺陷的扩展进行实验分析，可得到较为简单的 5 类存在安全缺陷的动作序列定义，如图 3-7 所示。图中，红色虚线范围内为开启本地进程的缺陷序列定义（Process launching vulnerability sequence），其具体描述为：

① 扩展调用 Browser core 动作类中的接口，执行 browser core 动作提供的功能。

② 扩展执行 accessibility 动作类提供的访问权限相关的功能，帮助扩展获得访问权限。

③ 调用 parser 动作类中的接口，完成 parser 动作类中的动作。

④ 通过 JavaScript core 引擎先关功能的调用完成 JavaScript 解释工作。

⑤ 通过 file access 动作类获得文件访问权限，并访问文件，经过此步骤，浏览器扩展可以通过接口调用完成本地文件和所有浏览器有访问权限的文件的访问，为开启本地进程或者完成其他功能做好充足准备，是影响扩展动作序列是否存在安全缺陷的重要因素。

⑥ 通过前面的铺垫和准备，利用 Process launching 动作类中的接口实现开启进程的功能，动作序列中的这一步骤是非常重要的一个环节。

⑦ 最终，扩展可以通过 Thread management 动作类中的接口，达成其恶意目的，或者造成非直接恶意但是可以被攻击者恶意利用的缺陷。

这几种动作类别的顺序组合方式就直接或者间接地导致扩展中存在与“开启本地进程”相关的安全缺陷，也就是说这几个动作类别的顺序组合，就可以被定义成可以应用于扩展序列安全性监测的存在安全缺陷的动作序列定义。

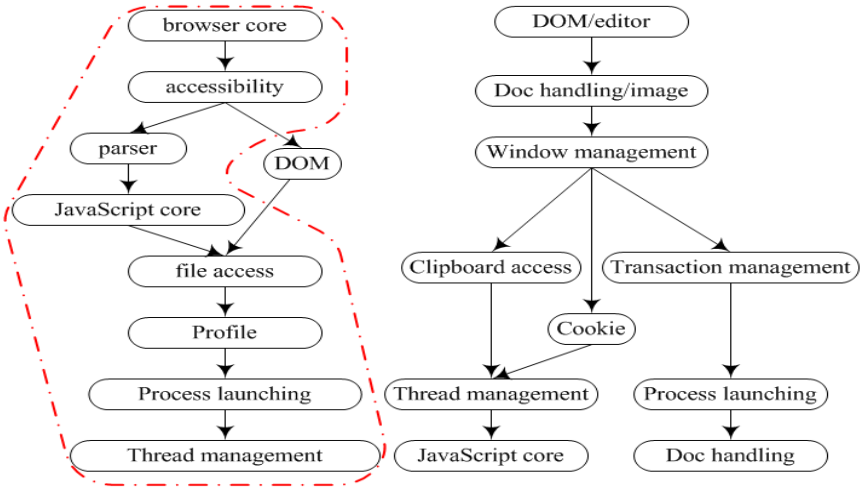


图 3-7 缺陷动作序列定义

图 3-7 中的其余几个序列的扩展动作序列的顺序组合方式分别为其余四种存在安全缺陷的动作序列定义，其中包括由 Browser core，accessibility，DOM，file access，Profile，Process launching，Thread management 组成的另一个开启本地进程的缺陷序列；由 DOM 或者 editor，Doc handling，Window management，Clipboard access，Thread management，JavaScript core 等动作类别组成，或者由 DOM，image，Window management，cookie，Thread management，JavaScript 等动作类别组成的与剪切板的访问相关的、存在安全缺陷的浏览器扩展动作序列。

上述的所有安全缺陷序列定义，在每两个相邻的动作类别之间，可以插入其余不在后续序列中出现的动作类，即每两个缺陷序列定义中相邻的动作类之间可以插于不会使得序列提前结束的动作类别名称，如：开启本地进程缺陷定义中 accessibility 和 DOM 之间插入任何非 file access，Profile，Process launching 或者 Thread management 的动作类别名，均不会破坏定义完成性和正确性。

3.3.2 动作序列检测

经过 3.2 节中所述方法约简后，可得到待检测扩展的有向无环行为序列图，对于每个待检测扩展的行为序列图，将其每个分支都作为一条输入，分别与所有预定义的缺陷序列匹配，即对所有拆分的扩展行为序列迹作为动作序列匹配工具的输入。如果对于任意一组输入，可以使得某个特定的 FSM 转换到 finish 状态，则匹配成功，报告相应类别缺陷；若全部输入没有任何匹配成功结果的返回，则判定扩展的行为是安全的。动作序列匹配部分中序列迹的检测方法如图 3-8 所示。

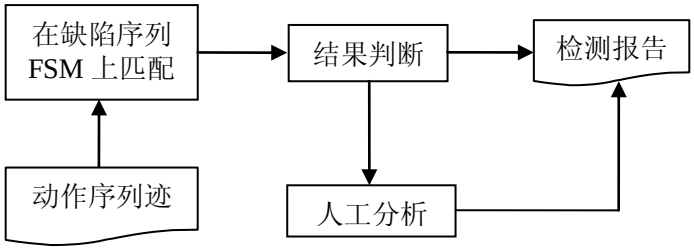


图 3-8 动作序列迹匹配方法

根据匹配结果的不同，应该采取不同的结果处理和报告方式。如果存在扩展的动作序列迹与预定义的缺陷序列完全匹配，则直接报告缺陷，可以忽略人工审查步骤；如果不存在部分匹配或者完全匹配，则可以直接判定扩展为安全扩展，也无需人工审查；如果出现部分匹配，则动作序列匹配工具不能判断扩展的安全性，需要进行人工审查，以确认缺陷或者排除正误识。因此扩展动作匹配流程可以描述为如图 3-9 所示。

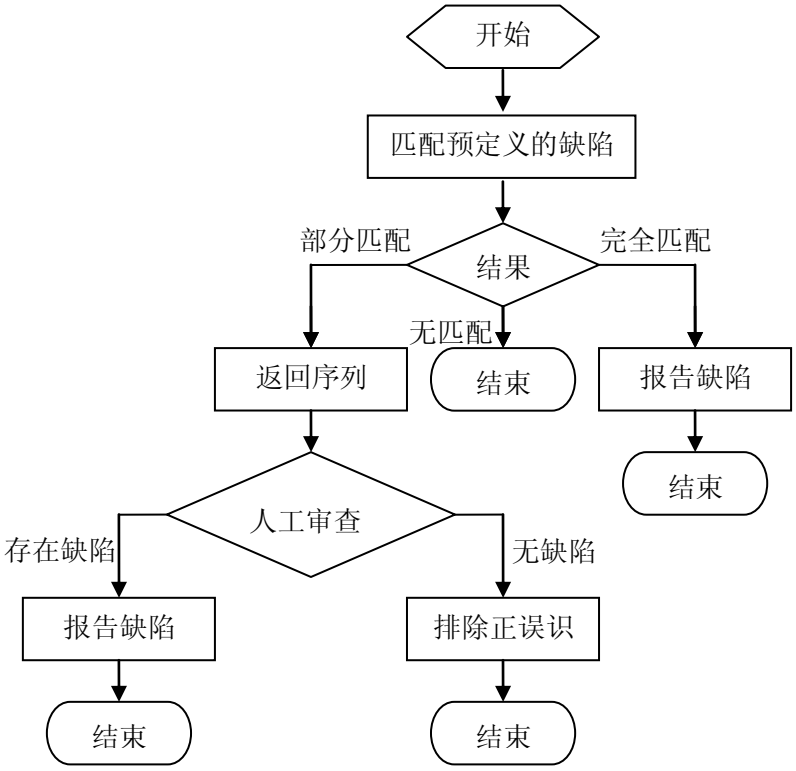


图 3-9 动作序列迹匹配流程

由于缺陷序列的定义还不够完善和精确，对于检测报告中认为存在缺陷的扩展，为排除正误识，还需要对其进行人工审查来最终确定安全性。

第四章 浏览器扩展安全检测工具的实现

浏览器扩展行为测试系统的实现依赖于多个工具及库，系统中各个模块在实现上也有诸多不同。本章将首先介绍系统实现中依赖的工具和库，然后对各个模块的实现进行详细阐述。浏览器扩展安全监测工具主要包括两部分，浏览器扩展监测系统和浏览器序列安全性检测工具。浏览器扩展监测系统的实现需要完成两部分重点内容，分别为原生浏览器的插桩工作和接口信息处理工作；而扩展序列检测工具的实现需要完成的主要工作是存在缺陷的动作序列 FSM 的定义，以及扩展动作序列迹的匹配工具的实现。

扩展监测系统负责实时监测扩展对 XPCOM 接口的调用信息，并根据单个接口的调用信息简单分析扩展安全性，继而提取扩展对接口的调用序列，然后完成接口调用序列的化简以及动作序列的映射以及扩展动作报告生成的任务。扩展运行时接口调用信息的抽取工作由插入浏览器源码中的钩子代码完成，接口信息处理则按照第三章中所述方法实现，两部分集成为扩展监测系统。

在扩展监测系统的输出结果之上，可以借助存在安全缺陷的扩展动作序列定义和动作序列迹匹配工具的支持，进一步对扩展的行为，也即扩展的动作序列做出安全性判断。换言之，对于一些在行为监测系统中不能发现安全缺陷的扩展，并不能排除其存在安全缺陷的可能性，需要对其动作序列进行检测，从而判断其安全性。本文中利用缺陷行为序列定义，查找被检测扩展中的安全缺陷。在缺陷序列匹配过程中，查询待检测扩展的动作序列中是否存在缺陷动作序列的匹配，并将匹配结果作为判断扩展安全性的依据。

4.1 扩展行为监测系统实现

本系统在实现上主要依赖的工具及库有：Firefox3.5、Libcurl[37]、Htmlcxx[38]以及 Sqlite[39]。完成扩展行为监测系统需要完成 Firefox3.5 的插桩以及接口处理模块。下文将对两部分的实现做详细的阐述。

4.1.1 浏览器插桩

接口调用信息包括 XPCOM 接口信息（接口名，参数，调用函数等），以及接口的调用序列。根据单个接口的调用信息以及接口所对应的安全等级划分，可

以在扩展行为监测阶段初步判断扩展的安全性：若可以判定存在潜在的安全缺陷，则可以直接给出扩展缺陷报告；反之，若不能判断是否存在安全问题，则需要对扩展对 XPCOM 接口调用的序列进行约简处理，简化为扩展行为序列，并在预先定义好的存在指定缺陷扩展行为序列定义的指导下，进一步通过匹配方法，判断待检测扩展的安全性。接口调用序列的约简，存在缺陷的扩展行为序列定义以及行为序列匹配方法的具体实现将在后文具体阐述。

为监测浏览器扩展行为，本文选择插桩浏览器的方法，对 Firefox 源码进行修改。本文主要通过编写钩子文件，并将钩子植入 Firefox 中的 JavaScript 引擎以及 XPConnect 模块（JavaScript 与 XPCOM 组件的桥梁），来获知扩展动态调用的 JavaScript 函数和访问的 XPCOM 接口信息，并在监测扩展运行过程中，控制插桩浏览器访问指定页面，同时模拟用户输入，即给出特定的扩展运行场景，方便记录和抽取扩展运行时接口调用信息以及接口访问序列。插桩浏览器时，钩子代码插入位置为 DOM 和 Javascript Engine 之间和 Javascript Engine 与 Parser 之间，用于监测扩展通过 XULRunner 和底层 XPCOM 接口之间的调用信息。钩子的实现方案如图 4-1 所示。

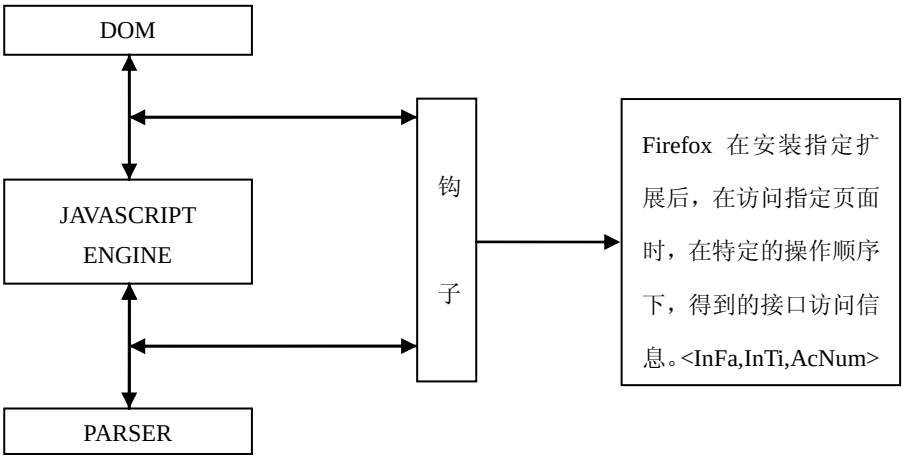


图 4-1 钩子程序实现方案

插桩浏览器模块是基于 Firefox 3.5 开源浏览器实现的。在 Firefox 中，XPConnect 模块主要负责 JavaScript 与 XPCOM 组件的交互。它作为二者之间的桥梁，实现在 JavaScript 与 XPCOM 组件进行通信时的相关转换工作，如参数类型转换等。通过在该模块插入相关的钩子，当浏览器扩展访问 XPCOM 接口时，系统会截获扩展所访问的接口、方法、甚至提取出访问方法时传递的参数。在 DOM 以及 content 模块中，通过向其中插入相关的钩子，当扩展填加或删除事件监听器时，系统会截获该信息，并获取事件监听器填加的位置以及监听的元素。浏览器通常通过注册事件监听器来与浏览器核心以及用户进行交互，而这一行为很有可能被间谍软件利用来盗取用户的隐私信息。与此同时，本文还对 content

模块做了进一步的修改以改变扩展中 overlay 文件的加载过程。当浏览器创建 overlay 文件中的一个元素时，系统会额外地为其填加一个“rootDocument”属性，并将其值设置为该扩展的标识符。这就使得扩展信息提取模块可以通过该属性来判断该元素是否是由扩展填加的。系统同时也在 Spidermonkey[40]（Firefox 中的 JavaScript 引擎）中进行了插桩。JavaScript 引擎通常提供对 JavaScript 语言最基本的支持，而用 JavaScript 实现的浏览器扩展中的大部分关键特性（如访问 DOM、访问网络等）则是由宿主应用程序（即 Firefox 浏览器）提供的。即使如此，JavaScript 引擎所提供的一些本地对象对于扩展行为的刻画仍然是十分重要的。例如 JavaScript 引擎提供的 eval 函数，它可以表明扩展具有动态代码执行的能力。通过在 Spidermonkey 中插入相关的钩子，系统可以截获扩展对这些关键 JavaScript 对象的访问。此外，本文还对 Firefox 的其他模块做了修改以简化扩展的安装过程，如 toolkit 模块以及 xpinstall 模块。

本文主要对其中的四个模块进行了插桩来截获浏览器的内部事件：XPConnect 模块、DOM 模块、content 模块以及 JavaScript 引擎。通过对 JavaScript 引擎以及 XPConnect 模块源码的分析，编写用于接口调用信息提取的钩子文件，将这些钩子插入到源码的上述位置，从而实现 Firefox 与监测模块的交互，完成浏览器的插桩工作。表 4-1 列举了几个可以实现扩展对 XPCOM 接口调用信息抓取的钩子，以及它们嵌入 Firefox 源码中的位置。其中，XPConnect_FilterXPCOMRequest 主要截获扩展对 XPCOM 接口进行的访问；js_BuiltInFunctionCalled 主要截获扩展对 JavaScript 内置函数的调用，dom_AddEventListener 则用于截获扩展在网页或 Firefox 界面中添加事件监听器的行为。

表 4-1 钩子及插入点

钩子	插入点
js_hooks.c: XPConnect_FilterXPCOMRequest(...)	①js/src/xpcconnect/src/xpcwrappednative.cpp: XPCWrappedNative::CallMethod(...) ②js/src/xpcconnect/src/qsgen.py
js_hooks.c: js_BuiltInFunctionCalled(...)	①js/src/jsobj.cpp: obj_eval() ②js/src/jsinterp.cpp:js_InvokeConstructor() ③js/src/jsinterp.cpp:js_Interpret()
dom_hooks.c: dom_AddEventListener(...)	①dom/src/base/nsDOMClassInfo.cpp: nsEventReceiverSH::AddEventListenerHelper() ②dom/src/base/nsDOMClassInfo.cpp: nsEventReceiverSH::RegisterCompileHandler() ③content/base/src/nsGenericElement.cpp: nsGenericElement::AddScriptEventListener()

钩子文件的植入时需要改写的源文件示例如图 4-2，表 4-1 中的 js_hooks.c 中的 XPConnect_FilterXPCOMRequest(...)函数调用点在目录 js/src/xpcconnect/src/ 中的 xpcwrappednative.cpp 文件的 XPCWrappedNative::CallMethod(...)函数中，为插入钩子和实现钩子 js_hooks.c 中函数的功能，需要将改写 xpcwrappednative.cpp 文件中的 XPCWrappedNative::CallMethod(...)函数，即在其中插入代码，处理上下文信息，收集调用钩子函数所需的信息和参数，并将钩子中函数的调用添加在源文件的适当位置，完成用于信息抽取的监听器的植入。xpcwrappednative.cpp 重新编译后，可以在扩展通过 XPConnect 调用 XPCOM 接口时完成部分监测和信息抽取功能。

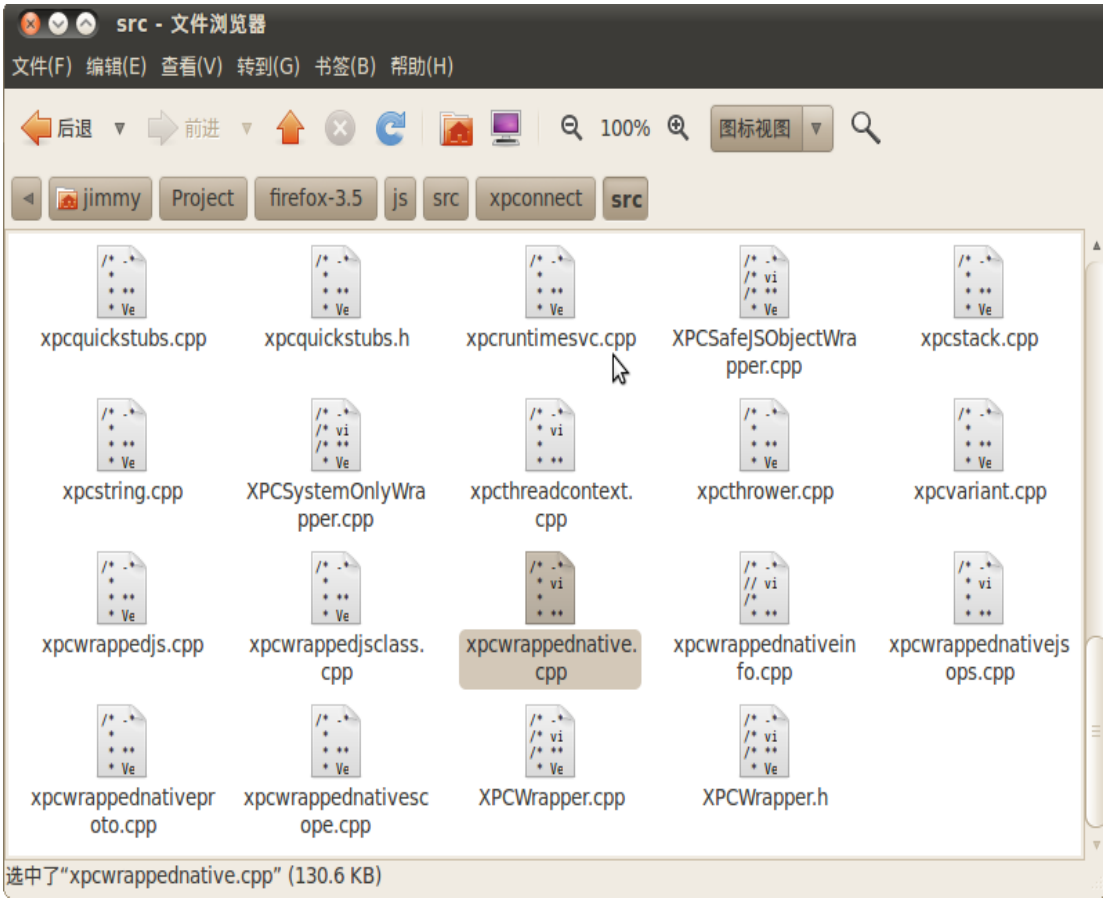


图 4-2 插桩浏览器时需要改写的源文件

在上述的所有钩子中，系统将截获的信息（如扩展访问了哪些接口等）存入了数据库以备分析。此外，为了能够获取扩展的间接行为，系统还实现了注入对象跟踪模块以追踪扩展插入到网页中代码的间接行为。在“跟踪”机制的实现中，本文首先识别出扩展往网页中注入代码时必须调用的多个入口函数（如 appendChild、insertBefore、document.write 等）。然后在这些函数相应的钩子中通过对调用参数的分析，将扩展与注入代码的依赖关系提取出来并写入数据库。例如，如果一个扩展通过调用“elem.appendChild(insertedElem)”动态地往网页

中注入了一段 JavaScript 脚本，则在 appendChild 对应的钩子中，系统会通过检查 elem 元素的 document 属性来确定扩展正在往哪个网页中注入脚本，同时通过分析 insertedElem 元素也可以获得关于注入脚本的详细信息。基于这些信息，在分析阶段就可以确定扩展的间接行为。

通过插桩浏览器所截获的扩展对接口的调用信息如图 4-3 所示。图中所见部分为扩展 handytab 对 XPCOM 接口的调用信息，信息中包括扩展所属类别，扩展名称，当前 id 下调用的接口名称，调用当前接口的方法名，接口的调用模式等，其中调用的接口的方法和调用模式可用于扩展行为监测系统简单判断扩展的直接安全性。插桩浏览器对 handytab 扩展的监测过程中，抽取的接口调用信息数目大约为 2 万 2000 条，图 4-3 中仅展示了其中的非常小的片段，如图所示的信息是由插入 XPConnect 的钩子 js_hook.c 中的函数完成的，其中主要负责信息记录的函数为 XPConnect_FilterXPCOMRequest(...)。

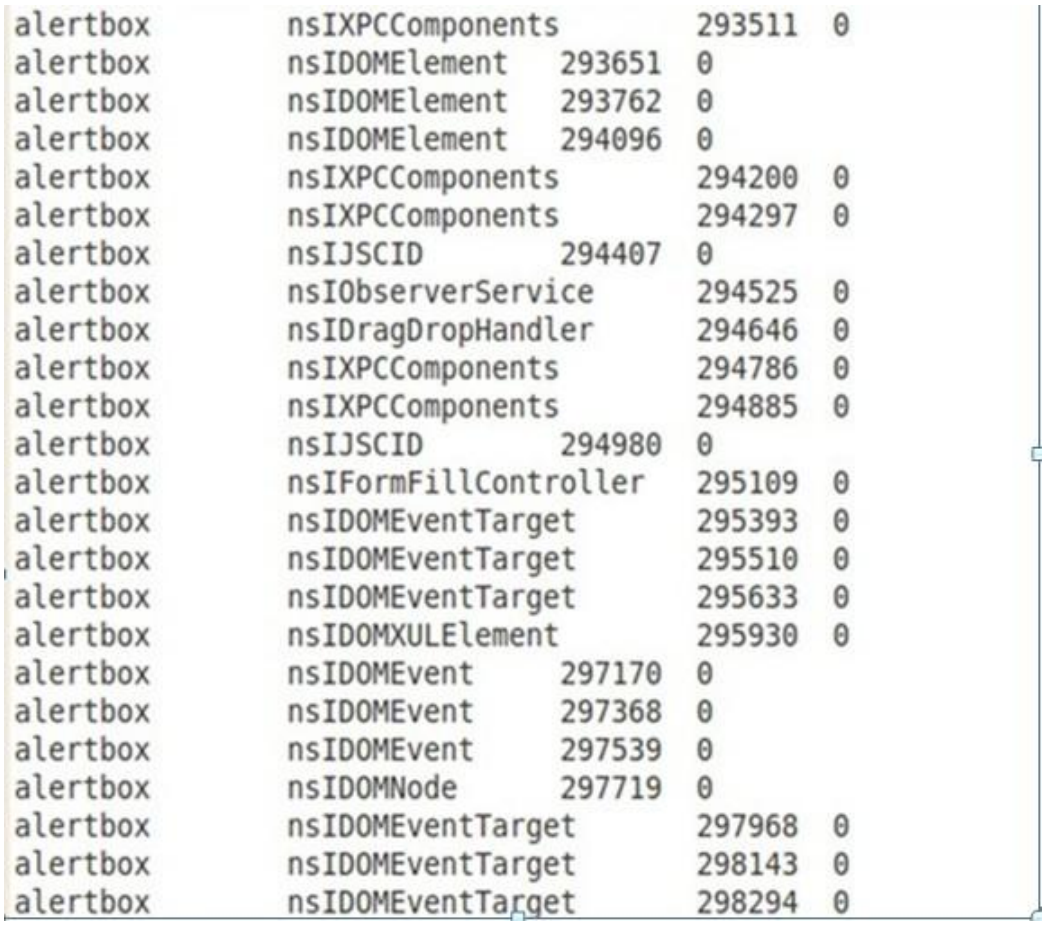
id	category	fullname	extensi...	interfacename	methodname	callmode	filename
70	appearance	HandyTab	handytab	nsIDOMElement	getAttribute	CALL_METHOD	chrome://handytab/content/handytab.js
71	appearance	HandyTab	handytab	nsIDOMElement	getElement...	CALL_METHOD	chrome://handytab/content/handytab.js
72	appearance	HandyTab	handytab	nsIDOMElement	hasAttribute	CALL_METHOD	chrome://handytab/content/handytab.js
73	appearance	HandyTab	handytab	nsIDOMElement	setAttribute	CALL_METHOD	chrome://handytab/content/handytab.js
74	appearance	HandyTab	handytab	nsIDOMEvent	target	CALL_GETTER	chrome://handytab/content/handytab.js
75	appearance	HandyTab	handytab	nsIDOMHTMLAn...	tagName	CALL_GETTER	chrome://handytab/content/handytab.js
76	appearance	HandyTab	handytab	nsIDOMHTMLBo...	ownerDoc...	CALL_GETTER	chrome://handytab/content/handytab.js
77	appearance	HandyTab	handytab	nsIDOMHTMLBo...	parentNode	CALL_GETTER	chrome://handytab/content/handytab.js
78	appearance	HandyTab	handytab	nsIDOMHTMLBo...	tagName	CALL_GETTER	chrome://handytab/content/handytab.js
79	appearance	HandyTab	handytab	nsIDOMHTMLDo...	body	CALL_GETTER	chrome://handytab/content/handytab.js
80	appearance	HandyTab	handytab	nsIDOMHTMLDo...	getElement...	CALL_METHOD	chrome://handytab/content/handytab.js
81	appearance	HandyTab	handytab	nsIDOMHTMLDo...	title	CALL_GETTER	chrome://handytab/content/handytab.js
82	appearance	HandyTab	handytab	nsIDOMHTMLFor...	tagName	CALL_GETTER	chrome://handytab/content/handytab.js
83	appearance	HandyTab	handytab	nsIDOMHTMLHt...	tagName	CALL_GETTER	chrome://handytab/content/handytab.js
84	appearance	HandyTab	handytab	nsIDOMHTMLInp...	parentNode	CALL_GETTER	chrome://handytab/content/handytab.js
85	appearance	HandyTab	handytab	nsIDOMHTMLInp...	tagName	CALL_GETTER	chrome://handytab/content/handytab.js

图 4-3 插桩浏览器获取的接口调用信息

4.1.2 扩展信息提取

扩展行为监测平台获取的扩展对接口的调用信息可以为扩展行为监测系统对扩展的安全性的初步判断提供数据支持，但是在后续的扩展的动作序列的安全性检测方法中，不需要方法名，调用模式，调用参数和扩展类别等信息，但是需要获取扩展对 XPCOM 接口的调用序列信息，即需要扩展名，调用接口名，调用时间和调用当前接口的用户行为编号，<InTi，InFa，AcNum>。如图 4-4 所示，

第一列为所检测的扩展名，示例中为火狐扩展 alertbox，后面三列分别为调用的接口名，调用时间，以及调用当前接口的用户行为编号。



```
alertbox        nsIXPCComponents        293511  0
alertbox        nsIDOMElement   293651  0
alertbox        nsIDOMElement   293762  0
alertbox        nsIDOMElement   294096  0
alertbox        nsIXPCComponents        294200  0
alertbox        nsIXPCComponents        294297  0
alertbox        nsIJSCID        294407  0
alertbox        nsIObserverService      294525  0
alertbox        nsIDragDropHandler      294646  0
alertbox        nsIXPCComponents        294786  0
alertbox        nsIXPCComponents        294885  0
alertbox        nsIJSCID        294980  0
alertbox        nsIFormFillController   295109  0
alertbox        nsIDOMEventTarget       295393  0
alertbox        nsIDOMEventTarget       295510  0
alertbox        nsIDOMEventTarget       295633  0
alertbox        nsIDOMXULElement        295930  0
alertbox        nsIDOMEvent     297170  0
alertbox        nsIDOMEvent     297368  0
alertbox        nsIDOMEvent     297539  0
alertbox        nsIDOMNode      297719  0
alertbox        nsIDOMEventTarget       297968  0
alertbox        nsIDOMEventTarget       298143  0
alertbox        nsIDOMEventTarget       298294  0
```

图 4-4 插桩浏览器获取的接口调用信息

图 4-4 为扩展行为监测系统中接口处理模块获取的初始序列信息，主要是用户行为编号为 0 的用户行为所引发的 alertbox 对 XPCOM 的接口的调用，主要为动作类别 DOM 和动作类别 XPConnect 中的接口，如 nsIJSCID，nsIXPCComponents，nsIDOMEventTarget，nsIDOMEvent 等。图中所示是，是植入浏览器的钩子 js_hook.c 中的函数获取的调用序列相关的信息。这一序列相关的信息表将传递给扩展行为监测系统中的接口序列处理模块。在接口调用的序列相关的信息中，许多接口名会在非常短的时间间隔内连续出现，这样就造成了信息的冗余，需要约简，以便于后续的接口序列的安全性分析和检测。实验中的大多数扩展，在未约简之前的接口调用信息大概为 2 万条。

4.1.3 接口调用信息处理

接口调用信息处理模块的主要任务是约简扩展的接口调用序列，并完成接口名到扩展动作类别的映射，以及映射后的动作序列的约简任务。因此，首先应该

化简原始的接口调用序列。由于接口调用序列中会出现大连的连续重复出现的接口名，因此需要序列进行去重处理。去重处理的方法为：对于序列中连续重复出现的接口名，保留第一个，同时计数，得到的结果中包括的信息为接口名以及其连续出现的次数。扩展 alertbox 的接口序列信息经过去重处理之后的结果如图 4-5 所示。图中 nsIDOMEvent 连续出现 27 次，去掉重复项 26 个，序列中保留一个，统计并记录其连续出现次数。

```
nsIDOMDocumentXBL       1
nsIDOMHTMLInputElement  1
nsIDOMElement   27
nsIDOMEvent     1
nsIDOMEventTarget       1
nsIDOMEvent     1
nsIDOMDocument  1
nsIDOMElement   1
nsIDOMDocument  3
nsIDOMHTMLHeadingElement        5
nsIDOMDocument  1
nsIDOMElement   1
nsIDOMDocument  1
nsIDOMElement   1
nsIDOMHTMLHeadingElement        9
nsIDOMHTMLDivElement    9
nsIDOMHTMLBodyElement   9
nsIDOMHTMLHtmlElement   9
nsIXPCComponents        2
nsIJSCID        1
nsIDOMElement   1
nsIStringBundleService  1
```

图 4-5 去重处理后的 alertbox 接口序列信息

接口调用序列中还会出现图 4-6 中所示情形，即几个接口以相同顺序连续出现，如图中的 nsIDOMDocument 和 nsIDOMXULElement 接口，对于此种情形，需要将连续出现的接口名捆绑，然后去掉后续相同顺序的接口名，以化简接口调用序列。即如图中所示的序列片段的化解结果应该为：只记录一次的 nsIDOMDocument 和 nsIDOMXULElement。

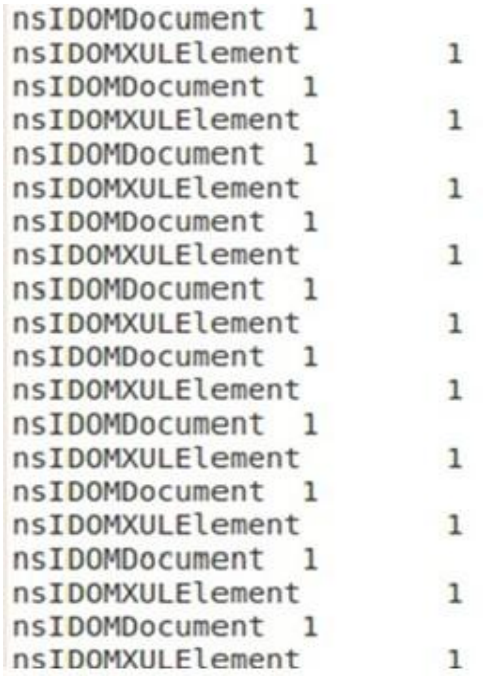


图 4-6 接口名循环交替出现的情形

对于去重之后的序列，按照各个接口所属的附录 A 中的接口的动作类别，对接口序列进行进一步处理，即将接口名序列映射为扩展动作类别序列。由于接口名和动作类别的对应关系为多对一，所以经过接口名到动作列别的映射之后，动作序列中会出现连续重复出现的动作类别，也因此需要进一步化简动作序列。但是在接口序列到动作序列的映射过程中即可完成动作序列中的重复项化简处理，即对于连续出现的多个对应到相同动作类别的不同接口，只记录一次动作类别，并统计该动作类别连续出现的次数。处理过程如下：

```
1. for each interface_name in the interface sequence
2.         look up the behavior group it belongs to;
3          if  interface_name in a behavior group
4.                 replace the current interface_name with the group_name;
5.                 if next interface_name in the same group;
6.                         delete next interface_name;
7.                 else  continue;
```

扩展对 XPCOM 接口调用序列通过上述过程转化为扩展的动作序列，经过映射的 alertbox 动作序列如图 4-7 所示。对于得到的动作序列，因为其中会包含连续交替出现的动作类别（如 alertbox 动作序列中的 DOM 和 DocHandling），仍需要按照接口序列中去掉重复项的方法进行约简处理，以期得到简化且并不丢失动作序列信息的扩展动作序列。

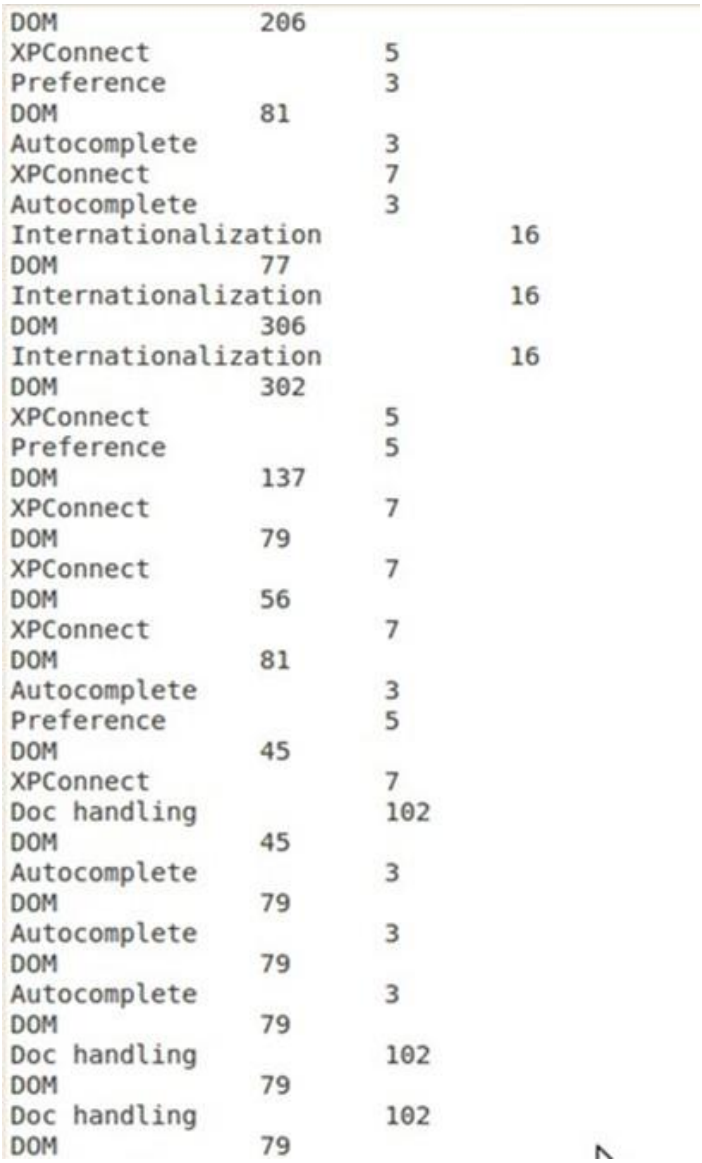


图 4-7 alertbox 的动作序列

在将扩展动作序列进一步化简的基础上，需要将简化的扩展动作序列转化为有向无环图（树状结构的序列），方法如下：

1. for every node(group in sequence) adding to the graph
2. if the next below the current node exists in current branch
3. check the sequence below the next;
4. if same with previous
5. ignore;
6. else add a new branch from the latest matched previous node;

4.1.4 扩展动作报告模块

经过对接口序列的处理和统计，以及对动作序列的分析和统计，给出所检测的扩展的动作报告。动作报告中包括当前检测的扩展的名称，如图 4-8 中所示，

扩展名为 alertbox。根据接口序列的处理和统计结果可得到 alertbox 调用的 XPCOM 接口总数为 223 个，并且根据未约简的接口序列直接映射得到的动作序列可统计出动作类别出现的次数最多的前三个，分别为：XPConnect，Data types and structures，DOM 三个扩展动作类别。并且根据最终的扩展动作序列，报告出扩展主要的行为（动作序列图中最长的分支），即主要动作序列：XPConnect，Additional xpcom services，Other tools，Data types and structures，Preference，Update，RSS/RDF，File access **，Component management，Internationalization 等。

在前述内容基础之上，给出扩展的可能的功能分析，alertbox 扩展动作报告中给出该扩展可能是用于访问 Dom，且该扩展可能会改变火狐浏览器的 UI，或者提供文件访问功能。根据扩展监测系统对扩展的动作分析给出初步安全性判断结果，报告缺陷，或者报告无直接缺陷。扩展动作报告中的缺陷判断的根据为一些简单的误用，如函数 eval()以及 overlay 的不正确应用等。因为监测系统主要用于扩展对接口调用信息的获取，所以对于可以不通过动作序列确定的直接缺陷的具体判断条件有待进一步改进。

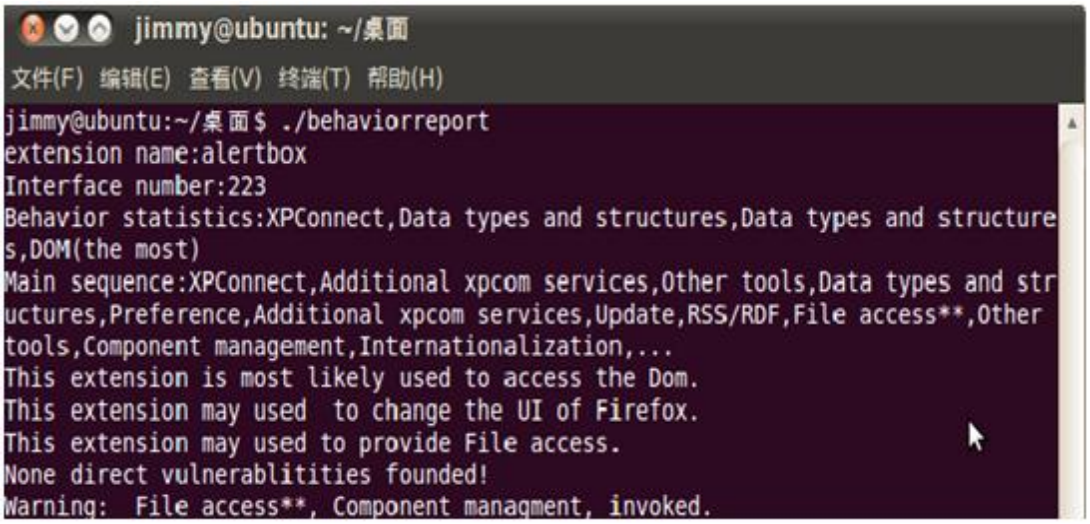


图 4-8 扩展 alertbox 的动作报告

4.2 动作序列匹配工具实现

对于存在直接安全缺陷的扩展，扩展行为监测系统可以检测并报告其缺陷；对于一些可能存在缺陷，却不能通过行为监测系统发现的扩展，需要进一步通过其动作序列判断安全性。动作序列的检测过程即为匹配扩展行为监测系统输出的动作序列迹和缺陷序列定义的过程，故而可以先构造缺陷序列定义的 FSM，继

而将扩展的所有动作序列迹作为缺陷序列 FSM 的输入，以其跳转的最终状态判断匹配程度。缺陷扩展动作序列 FSM 和序列匹配模块将在后文详细阐述。

4.2.1 缺陷扩展动作序列 FSM

根据 3.4.1 节中的存在缺陷的扩展的动作序列定义，构造如下图所示的缺陷序列 FSM，用以匹配待检测扩展的动作序列迹。在 Process launching 缺陷 FSM 中，如图 4-9 所示，如果序列迹中不存在动作类别 Browsercore，状态机将一直停留在 start 状态不做任何状态的跳转。

若其接受 Browsercore 输入，将跳转到状态 1，此时，对于非动作类别 Accessibility 的其余任何输入，状态机不进行任何状态转移；若输入为状态类别 Accessibility，则跳转到状态 2。

在状态 2 下，当前输入为动作类别 Parser，则跳转到状态 3；如果当前输入为动作类别 DOM，则转移到状态 5；对于其余输入，则保持状态不变。

在状态 3 下，当前输入为任意非动作类别 Javascript core 时，状态机保持在状态 3；只有当前序列迹的输入为动作类别 Javascript core 时，状态机完成从状态 3 到状态 4 的状态转移。

在状态 4 或者状态 5 下，只有接收到收入为动作类别 File access 时才能够跳转到状态 6。如果在状态 6 下当前输入为动作类别 Profile，则转移到状态 7；若当前输入为其他动作类别名称，则进入 exit 状态，说明匹配结果为部分匹配，且匹配过程终止。

如果动作序列迹使得 Process launching 缺陷状态机进入到状态 8，且动作序列迹的后续输入中存在动作类别 Thread management，则状态机会进入 finish 状态，即当前动作序列迹与 Process launching 缺陷序列完全匹配，所检测的扩展存在此类安全缺陷。

如果当前动作序列迹不能使状态机到达 exit 状态或者 finish 状态，且输入已到达动作序列尾，则匹配结果为不匹配，当前的动作序列迹的匹配工作结束。

对于匹配结果，若扩展的所有动作序列迹中存在某条输入可以使得其与缺陷动作序列的匹配结果为完全匹配，说明被检测的扩展有存在此类安全缺陷；如果所有的动作序列迹输入均得到不匹配的结果，则说明当前扩展不存在此类安全缺陷；如果扩展的所有动作序列迹的匹配结果存在完全匹配，且不存在完全匹配，则说明所检测的扩展有存在此类安全缺陷的可能性，但其安全性不能准确通过匹配工具判断，为排除正误识，此时需要借助扩展动作报告和人工审查结合分析。

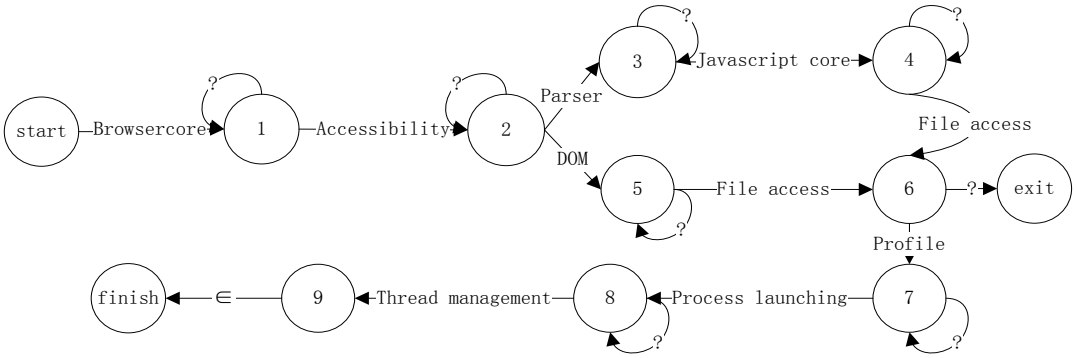


图 4-9 Process launching 缺陷 FSM

与 Process launching 缺陷 FSM 的状态转移方式类似，剪切板相关的缺陷，以及 Cookie 相关的安全缺陷和事务处理先关缺陷的自动机如图 4-10 所示。其中在状态 5 下可以接受非动作类别 Thread management 的动作类别名称，从而跳转到部分匹配状态 exit。如果状态机转移到状态 7 或者状态 8，则其直接自动跳转到完全匹配状态 finish。其中每个？都代表除能够使当前状态转移至下一状态的输入动作类别的其他动作类别名称。

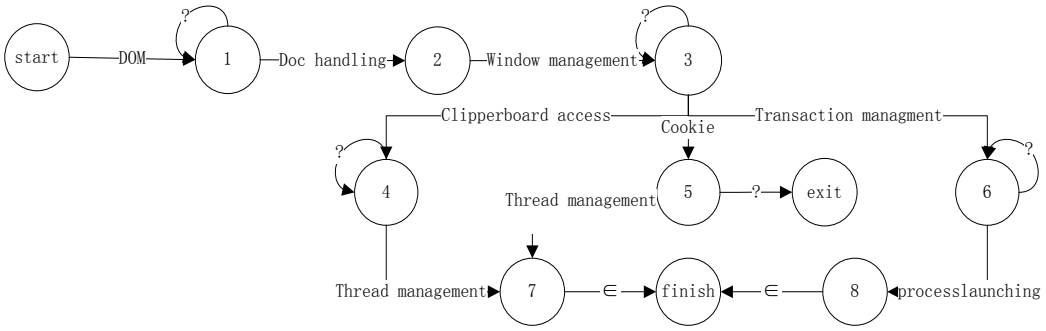


图 4-10 剪切板、Cookie 相关的缺陷 FSM

在缺陷序列定义应用 FSM 实现后，序列匹配模块只需要接收扩展行为监测系统所输出的扩展动作序列迹，并分别让缺陷序列自动机接收输入，以得到匹配结果。

4.2.2 序列匹配模块

序列匹配模块负责接收记录扩展动作序列迹的 txt 格式的输入文档，每读取一行数据（一行数据为一条动作序列迹，动作类别名称之间以水平制表符隔开），分别将其与所有缺陷序列定义匹配，即检测所有的动作序列迹的安全性。由于缺陷序列的定义中，序列中的每个动作类别名称与扩展动作监测平台获取的扩展动

作序列迹中的动作类别名称有着严格的一致性，匹配中则只需要比较文档中当前的字符串与缺陷序列 FSM 中的状态转移条件是否一致，即可完成匹配过程。

图 4-11 为扩展 alertbox 的动作序列迹与 process launching 缺陷序列的匹配结果，其动作序列中的 8 个分支均不能使该缺陷序列 FSM 转移到 exit 或者 finish 状态，即不存在部分匹配或者完全匹配，此时则可以断定扩展 alertbox 中不存在 process launching 缺陷。

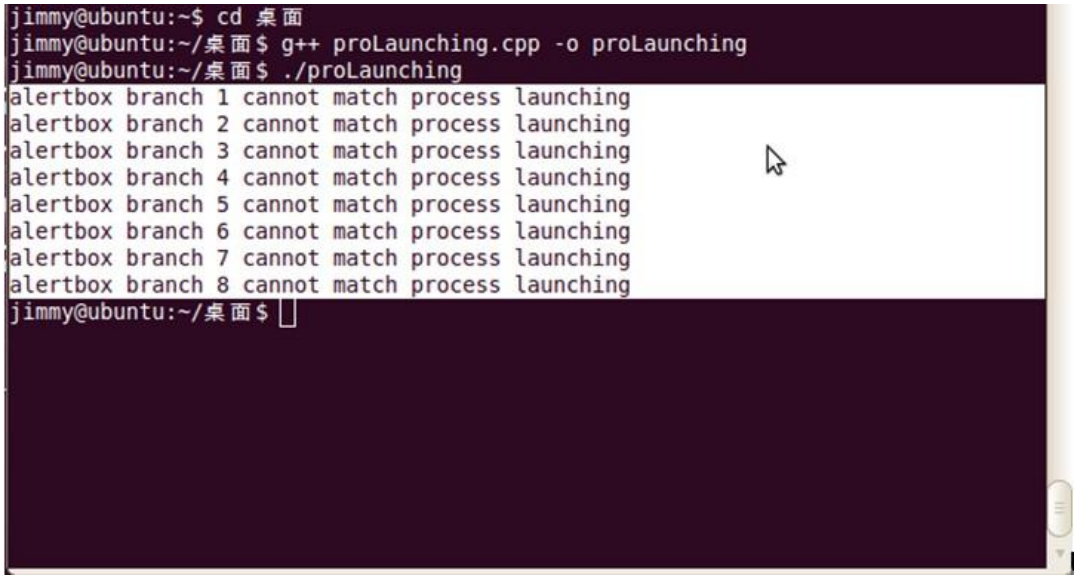


图 4-11 扩展 alertbox 的动作序列迹匹配结果

第五章 实验分析

5.1 有效性实验

在行为监测系统和行为序列匹配工具的支持下，针对表 3 中的四个存在安全缺陷的 Firefox 扩展[42]，构造和还原攻击场景，在 firefox 下做攻击实验，并应用前文所述扩展缺陷检测方法，分别对下表中的扩展进行了测试，实验结果如下：

表 5-1 有效性测试结果

实验	涉及的扩展	Firefox	缺陷检测结果
1	CoolPreview V2.7.2	被攻击	在行为监测系统下报告缺陷
2	Yoono V6.1.0	被攻击	在行为监测系统下报告缺陷
3	ScribeFire V3.4.1	被攻击	在行为监测系统下报告缺陷
4	Feed Sidebar V3.1	被攻击	在扩展动作序列匹配工具下报告缺陷

从试验结果可以看出，对于四个存在缺陷的扩展，Firefox 3.5 浏览器全部被攻击成功，扩展行为监测系统则成功地报告了扩展中的安全缺陷，虽然第四个扩展则不能在扩展行为监测系统下发现安全缺陷，但是通过对第四次攻击场景的分析，不难发现，在整个攻击过程中扩展并没有对任何存在安全隐患的接口进行访问，只是通过对一系列安全接口的有序访问形成了一次攻击，从而使得扩展行为监测系统不能发现扩展缺陷。但是，在动作序列匹配方法指导下，序列匹配工具成功的在动作序列中发现能够使得 process launching 缺陷序列 FSM 转移到 finish 状态的扩展输入，从而检测到第四个扩展的缺陷。

实验结果一定程度上说明了行为监测系统与动作序列匹配工具在浏览器扩展的缺陷检测中结合使用的扩展缺陷检测方法的正确性，即不仅需要监测扩展访问的关键接口，还需将扩展访问接口的序列，也即扩展的动作序列作为扩展缺陷检测的一项依据。

5.2 性能评价实验

为了评估浏览器扩展行为监测系统在运行时的额外开销，本文选择记录插桩浏览器的响应时间延迟和内存开销，经过五十次的计算，取其平均值作为监测系统的性能测评数据。由计算结果可知，插桩后的浏览器在响应时间上和内存消耗上均多余原生的火狐浏览器，但是其性能与原生火狐浏览器差别并不大。

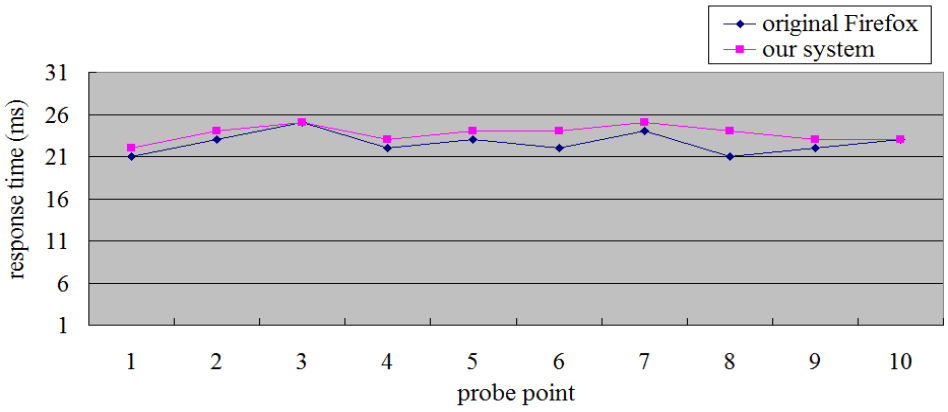


图 5-1 插桩浏览器与原生浏览器响应时间图

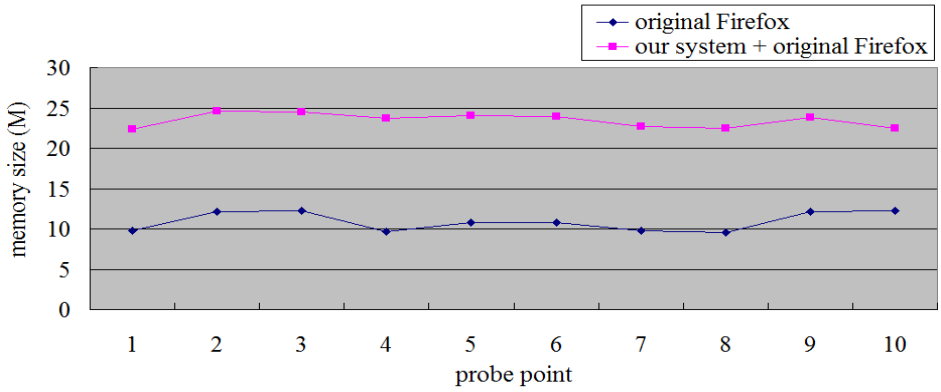


图 5-2 插桩浏览器与原生浏览器内存消耗图

图 5-1 显示插桩后的浏览器在响应时间上比原生浏览器稍慢，延迟时间比例为 4.31%，图 5-2 显示插桩后的浏览器在内存消耗上是原生浏览器的 2 倍。但是由于本文提出的浏览器扩展安全缺陷检测方法是异步的，不会影响用户对浏览器的使用，所以其性能上的损失是在容忍度以内的。

第六章 结论与展望

6.1 结论

当前，网络浏览器已经成为人们使用最多的应用程序之一，其本身也从用于解析和渲染静态页面的简单应用软件演变为支持众多 Web 标准的复杂系统软件。然而，浏览器逐渐强大的功能背后也隐藏了越来越多的安全漏洞，时时刻刻地威胁着用户的安全，特别是在引入扩展机制之后。

为了能够更好地了解目前浏览器扩展的行为，本文对 Firefox 现有扩展的行为进行了分析和研究，并通过对 Firefox 浏览器为扩展提供的接口进行分析，对 Firefox 扩展的行为进行了抽象及分类，识别出 57 种不同的 Firefox 扩展的动作类别并按照其对用户可能产生的威胁将其划分到 4 个不同的安全等级中。

本文对浏览器的扩展机制及其存在的安全问题进行了深入的研究，同时为了缓和这一安全问题，提出了基于行为监测分析的浏览器扩展安全缺陷检测方法，并且在检测方法的指导下，设计和实现了浏览器扩展的行为监测原型系统以及缺陷序列匹配工具，用以对扩展的行为进行实时监测并提取、分析并处理扩展运行时对 XPCOM 接口调用信息，以扩展缺陷报告或者扩展行为序列图作为监测系统的输出，并利用缺陷序列匹配方法，对扩展行为序列的安全性进行进一步判断。实验表明，该方法虽然给浏览器带来了一定的负载，但却可以作为一类有效的异步检测手段，判断扩展的安全性，降低浏览器扩展机制带个用户的安全风险

6.2 展望

同时，实验结果也为今后的研究工作指明了方向，即：需要更细致深入的对由合法的行为序列导致的扩展安全缺陷加以研究，以更加精确的缺陷序列定义，丰富和完善该扩展安全缺陷检测方法。

本文提出的浏览器扩展安全缺陷检测方法还存在诸多需要改进和改善的地方，具体包括如下几点：

- ✧ 在系统的目前实现中，虽然在扩展测试时会通过模拟一系列的用户行为来暴露更多的扩展行为，但是设计的模拟用户行为有限，且对接口的调用序列的重新组合方式有待进一步改进。

- ✧ 扩展行为监测系统目前只能监测到扩展访问的所有接口、方法信息以及一部分方法的参数信息。为了能够更好地刻画浏览器扩展的行为，在今后的工作中有必要对所有关键方法的参数信息进行提取，以及对扩展在运行时的数据流进行跟踪。
- ✧ 本文主要的分析对象为扩展访问的接口或方法序列，研究内容是监测和检测扩展的行为的安全性，目前对于扩展的不安全行为和缺陷行为序列的定义尚不完善和精确。因而下一步需要在大量研究和分析的基础上进一步精确对缺陷的定义。

参考文献

[1] Mozilla Foundation. Extensions[EB/OL]. (2011-04-04) [2011-05-07]. https://developer.mozilla.org/en/Extensions.

[2] D. Turner. Symantec Internet Security Threat Report: Trends for January-June. http://eval.symantec.com/mktginfo/enterprise/white_papers/ent-whitepaper_internet_security_threat_report_xii_09_2007.en-us.pdf. 2007.9

[3] LIVERANI, ROBERT S. Abusing Firefox Extensions [EB/OL].(2009-08-14) [2011-05-07]. http://security-assessment.com/presentation/all/all/archive.htm.

[4] Adobe. Flash player update available to address security vulnerabilities. http://www.adobe.com/support/security/bulletings/apsb07-12.html.

[5] P. D. Petrkov. Quicktime pwns firefox. http://www.gnucitizen.org/blog/0day-quicktime-pwns-firefox.

[6] AusCERT. Sun java runtime environment vulnerability allows remote compromise. http://www.auscert.org.au/render.html?it=7664

[7] VERDURMEN J. Firefox extension security[D]. Netherlands:Raboud University. 2008.

[8] Mozilla Foundation. Extensions[EB/OL]. https://developer.mozilla.org/en/Extensions.

[9] Kristjan Krips. The Security Analysis of Browser Extensions. Bachelor's thesis, University of Tartu, June 2010

[10] Security-Assessment.com. Cross Context Scripting with Firefox[EB/OL]. http://www.security-assessment.com/files/documents/whitepapers/Cross_Context_Scripting_with_Firefox.pdf

[11] Security-Assessment.com. Exploiting Cross Context Scripting Vulnerabilities in Firefox[EB/OL]. http://www.security-assessment.com/files/documents/whitepapers/Exploiting_Cross_Context_Scripting_vulnerabilities_in_Firefox.pdf

[12] BRENT S. XULRunner: A New Approach for Developing Rich Internet Applications[J]. IEEE Internet Computing, 2007, 11(3):67-73.

[13] PARRISH R. An introduction to XPCOM[EB/OL]. (2009-02-16) [2011-05-07].

[14] VERDURMEN J. Firefox extension security[D]. Netherlands:Raboud University. 2008.

[15] Mozilla Foundation. Xpcnativewrapper for extensions[EB/OL]. http://developer.mozilla.org/en/docs/XPCNativeWrapper.

[16] T. Amtoft and A. Banerjee. Information flow analysis in logical form. In R. Giacobazzi, editor, SAS 2004, volume 3148 of LNCS, pages 100–115. Springer-Verlag, 2004.

[17] Sruthi Bandhakavi, VEX: Vetting Browser Extensions For Security Vulnerabilities, 2010

[18] Adam Barth, Adrienne Porter Felt, Prateek Saxena, and Aaron Boodman. Protecting browsers from extension vulnerabilities. In Network and Distributed System Security Symposium, 2010.

[19] David M. Martin Jr., Richard M. Smith, Michael Brittain, Ivan Fetch, and Hailin Wu. The privacy practices of web browser extensions. Communications of the ACM, 2001.

[20] Mike Ter Louw, Jin Soon Lim, and V. N.Venkatakrishnan. Extensible web browser security. In Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA '07, pages 1{19, Berlin, Heidelberg, 2007. Springer-Verlag.

[21] SALVATORE G, BENJAMIN L. Gatekeeper: Mostly static enforcement of security and reliability policies for javascript code[C].USENIX Security Symposium, 2009.

[22] SRUTHI B, SAMUEL T K, MARIANNE W et al. Vex: vetting browser extensions for security vulnerabilities[C]. Proceedings of the 19th USENIX conference on Security, 2010.

[23] MOHAN D, VINOD G. Analyzing information flow in javascript-based browser extensions[C]. Computer Security Applications Conference, ACSAC, 2009.

[24] MIKE T L, JIN S L, VENKATAKRISHNAN V. Enhancing web browser security against malware extensions[J]. Computer Virology, 2008,4:179–195.

[25] MIKE T L, JIN S L, VENKATAKRISHNAN V. Extensible web browser security[C]. Proceedings of the 4th international conference on Detection of Intrusions and Malware,and Vulnerability Assessment, 2007.

[26] OYSTEIN H, GIOVANNI V. Detecting malicious javascript code in mozilla[C]. International Conference on Engineering of Complex Computer Systems, 2005.

[27] LI Zhuowei, WANG Xiaofeng, CHOI J. Spyshield: Preserving privacy from spy add-ons[C]. Proceedings of RAID, 2007.

[28] ANDRE C, BRYAN L. Conceptual Architecture of Firefox[EB/OL]. http://web.uvic.ca/~hitchner/assign1.pdf.

[29] Mozilla Foundation. XPInstall[EB/OL]. https://developer.mozilla.org/en/XPInstall.

[30] BRENT S. XULRunner: A New Approach for Developing Rich Internet Applications[J]. IEEE Internet Computing, 2007, 11(3):67-73.

[31] Nagamanoj Vankadhara. Plugin for Online Program Compiler [EB/OL]. http://www.cse.iitb.ac.in/~nagamanojv/cs699/IPStage1.html#(1).

[32] PARRISH R. An introduction to XPCOM[EB/OL]. http://www.ibm.com/developerworks/webservices/library/co-xpcom.html.

[33] Mozilla Foundation. XPCOM[EB/OL]. http://www.mozilla.org/projects/xpcom/.

[34] Mozilla Foundation. XBL[EB/OL]. https://developer.mozilla.org/en/XBL

[35] Ant. Firefox Extension Spyware. Hacks in Taiwan Conference 2008, http://hitcon.org/hit2008/download/HIT2008-Firefox_Extension_Spyware-Ant.pdf.

[36] Mozilla Foundation. Firefox 3.5 for developers[EB/OL]. https://developer.mozilla.org/en/Firefox_3.5_for_developers.

[37] libcurl - the multiprotocol file transfer library. http://curl.haxx.se/libcurl/.

[38] htmlcxx - html and css apis for c++. http://htmlcxx.sourceforge.net/.

[39] Sqlite. http://www.sqlite.org/.

[40] Spidermonkey. https://developer.mozilla.org/en/SpiderMonkey.

[41] Security-Assessment.com. CoolPreviews Firefox Extension Privileged Code Injection [EB/OL]. (2009-08-25) [2011-05-16].

[42]

发表论文和参加科研情况说明

发表的论文：

[1] Junjie Wang，Xiaohong Li，Qin Wang，Guangquan Xu“Characteristic Behavior Sequence Based Attack Detection Method for Browser Extension”，2012/07

[2] Qin Wang, Xiaohong Li, Bobo Yan “A browser extension vulnerability detecting approach based on behavior monitoring and analyzing”, ICETS，2012/09

授权的专利：

一种 Firefox 扩展的安全缺陷检测方法

参与的科研项目：

致　谢

本论文的工作是在我的导师李晓红教授的悉心指导下完成的，李晓红教授严谨的治学态度和科学的工作方法对我的学习和生活产生了极大的帮助和影响。在此衷心感谢李晓红老师近三年来热心的指导和关心。

冯志勇教授在我的科研工作和生活中也进行了悉心的指导和关怀，在我遇到困难时，冯老师的一句鼓励总是能使我很快找到前行的方向。同时，许光全副教授对我的科研工作也给予了很多悉心指导、提出了很多宝贵意见，使我的个人能力有了很大的提高。在此，一并向两位老师表示衷心的感谢。

在实验室工作及论文撰写期间，王建刚、王俊杰、余小飞、张倩倩等同学对我的研究工作给予了热情的帮助，在此向他们表达我的感激之情。此外，严波波等同学也对我的研究工作提出了很多宝贵的意见和建议，在此一并向他们表达感谢。

另外也感谢我的父母、亲人及所有朋友，他们的理解和支持使我能够在学校专心完成我的学业。

附录 A 57 个 XPCOM 接口行为类别划分

1) Accessibility，访问浏览器为辅助设备（如放大镜、屏幕阅读器等）提供的接口。

2) Browser core，访问浏览器内部核心接口。

3) Auto complete，用于实现自动补全功能。

4) Log to console，向控制台输出信息。

5) History，管理用户的浏览历史。

6) Password，管理密码信息。

7) Searching，用于实现搜索功能。

8) Spell checking，用于实现拼写检查功能。

9) Bookmark，管理书签。

10) Login，管理用户登陆信息。

11) Cookie，管理用户的 Cookie 信息。

12) Preference，管理用户的偏好信息。

13) Profile，管理用户的 Profile 文件。

14) DOM，访问 DOM。

15) Editor，使用编辑器。

16) File access，访问文件系统。

17) Internationalization，用于实现国际化功能。

18) Offline cache，管理离线缓存文件。

19) Parser，解析 XML 或 INI 文件。

20) Network access，访问网络。

21) Channels, 通道。

22) Protocol handlers, 协议处理。

23) Sockets, 网络 socket 通信。

24) Proxies, 代理。

25) Download, 下载。

26) Network Utilities, 网络公用程序。

27) RSS/RDF，处理 RSS、RDF 数据。

28) Data types and structures，访问 XPCOM 提供的基本数据类型及数据结果。

29) Streams，处理流。

30) Memory management，对内存进行管理，如分配、释放。
31) Process launching，启动进程。
32) Thread management，对线程进行管理。
33) Component management，管理 XPCOM 组件。
34) Additional XPCOM services，其他 XPCOM 提供的功能。
35) Javascript core，访问 JavaScript 相关的核心接口。
36) Javascript debugger，访问用于调试 JavaScript 的接口。
37) XPConnect，访问 XPConnect 模块提供的接口。
38) Authentication，用于实现授权功能。
39) Certification，用于实现认证功能。
40) Crptograph，用于实现加密。
41) Additional security interfaces，其他安全相关的行为。
42) Doc handling，文档的加载、渲染与显示。
43) Transaction management，管理事务。
44) Web worker，实现 HTML5 中的 Worker 技术。
45) Window management，管理窗口。
46) Clipboard access，访问剪切板。
47) Print，用于实现打印功能。
48) Other graphics and widgets，其他与图形相关的功能。
49) Addons management，管理 Addons。
50) XPInstall，安装扩展。
51) Update，更新应用程序。
52) Other tools，调用其他的实用工具。
53) Storage，访问 sqlite 数据库。
54) Images，处理图片。
55) ZIP/JAR process，处理 ZIP 或 JAR 文件。
56) JVM，访问 JVM。
57) Plugins，与插件交互。

浏览器扩展安全缺陷检测工具的设计与实现

作者： 王钦
学位授予单位： 天津大学