doi :10.3969/j.issn.1671-1122.2011.08.024

一个典型的 Web 安全评测工具的分析与改进

宋海龄，文伟平

（北京大学软件与微电子学院，北京 102600）

摘 要：文章较全面地分析和总结了现有的 Web 漏洞挖掘技术及工具，以开源的 Web 漏洞扫描工具 Paros Proxy 为研究对象，对 Paros Proxy 的爬虫模块及检测模块进行深入研究和分析，进而对其进行改进。经测试，改进后的 Paros 爬虫模块支持 JavaScript URLs 的解析及爬行，可以提取到更多的网页链接，而改进后的检测模块，在漏洞检测性能及效率上也有明显提高。

关键词：网络爬虫；漏洞检测；线程池；Web 应用

中图分类号：TP393.08 **文献标识码：**A **文章编号：**1671-1122（2011）08-0065-04

Analysis and Improvement of a Typical Web Security Assessment Tool

SONG Hai-Ling, WEN Wei-Ping

(*Department of Software Technology, SSM, Peking University, Beijing* 102600*, China*)

Abstract: This paper makes a comprehensive analysis and summary of the existing Web loophole mining technology and tools, to open source Web vulnerability scanning tool Paros Proxy as the research object, the Paros Proxy crawler module and a detection module for in-depth research and analysis, and its improvement. After the test, the improved Paros crawler module supports the JavaScript URLs analytical and crawling, can extract more webpage link, and the improved detection module, the vulnerability detection performance and efficiency can be improved significantly.

Key words: web spider; vulnerability detection; thread pool; web application

1 漏洞挖掘技术

目前，挖掘漏洞技术有很多种，本文主要介绍静态分析技术、Fuzzing 技术及动态调试技术。

1.1 静态分析技术

静态分析技术（Static analysis）包括源代码审核和二进制审核，是指不执行程序，通过静态分析汇编代码的方法来获取待检测的可执行代码内部信息。

目前，静态分析主要有类型推断、数据流分析和约束分析三种方法。在实际应用中，一般是将几种不同的静态分析方法结合起来，共同完成一项检测任务。例如，先以类型限定词的推断规则作为约束传播规则，进行约束传播，再利用标准的约束求解器发现程序中的格式化字符串漏洞。

但静态分析技术在某种程度上会存在一定检测问题。因为使用静态技术对程序执行时的状态和行为的集合进行预测，这其中包括的一些在程序运行时报出的错误是不可判定的，并且没有科学可靠的方法可以对此进行解释验证关于这个错误是否源自于程序本身，这在数学理论上是不可实现的。

1.2 动态调试技术

动态调试技术（Runtime analysis）[1] 是指在调试器环境中运行目标程序，并在运行时分析其反汇编代码，利用反汇编技术鉴别可能出现漏洞的代码序列。动态调试技术与静态分析非常相似，主要是观察分析目标代码而并不是对其实施逆向工程，也是一个在调试器（如 Ollydbg）帮助下进行的手动过程。使用动态调试技术进行漏洞挖掘的主要技术有“数据跟踪”（Data follow）和“代码跟踪”（Code follow）。但在实际检测中，由于动态测试只能对有限的测试用例进行检查，所以不能保证发现软件的所有安全漏洞。

收稿时间：2011-06-10

作者简介：宋海龄（1986-），女，北京，硕士研究生，主要研究方向：软件测试与质量保证；文伟平（1976-），男，湖南，副教授，博士，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等。

1.3 Fuzzing技术

Fuzzing是一种基于黑盒测试思想的漏洞发现技术。Fuzzing技术的有效性在被得到验证及推广后，很多领域纷纷引入了这种技术开发自己的Fuzzing工具。

与其它技术相比，Fuzzing技术可以说是最自动化的，测试效果也是最明显的。因为：1）其测试目标是二进制可执行代码，比基于源代码的白盒测试方法适用范围更广泛；2）Fuzzing是动态实际执行的，不存在静态分析技术中存在的大量误报问题；3）Fuzzing原理简单没有大量的理论推导和公式计算，不存在符号执行技术中的路径状态爆炸问题；4）Fuzzing自动化程度高，无须在逆向工程中投入大量的人力。

2 Paros Proxy 工具的研究与分析

2.1 Paros Proxy工具简介

Paros Proxy[2] 使用Java开发，是一款开源软件，与Nikto相比，Paros使用了更友好的图形界面、详细美观的报表，并且具有更多的功能。

Paros可以通过代理服务器的方式记录浏览器与Web服务器之间发生的所有请求和响应，并以适当的形式显示出来。还可以拦截浏览器和Web服务器之间传送的请求或响应，并可以对这些响应信息进行任意的修改。Paros内置的爬虫模块支持同时启动五个线程抓取，同时还支持通配符配置不需要抓取的页面名称或域名。另外，与Nikto的安全漏洞扫描工具类似，Paros的Web应用扫描模块通过发送一系列准备好的请求URL，分析服务器响应的内容，判断是否存在相关的安全漏洞。目前Paros的扫描插件包括信息收集、客户端浏览器、服务器安全、注入漏洞及其他等五类，每类又被细分为若干的小类，用户可以选择扫描所使用的不同插件。

2.2 Paros Proxy功能模块分析

Paros Proxy包括一个Web代理程序、Web爬虫（Spider）和Hash计算器，还有一个可以测试常见的Web应用程序攻击（如SQL注入式攻击和跨站脚本攻击）的扫描器，因此，它是一个典型的基于Web拦截代理的综合工具套件。

2.2.1 Web拦截代理

Paros Proxy内置的Web拦截代理需要设置两个监听端口，8080和8443，其中8080是代理连接端口，8443是SSL端口，通过对端口的通信监听，Paros能够截取和修改所有在服务器和客户机之间的HTTP和HTTPS数据，包括cookies和构成领域。

2.2.2 Web爬虫模块

Paros Proxy的爬虫模块采用广度优先爬行策略，并支持多线程采集。对于并行爬虫架构而言，空的队列并不意味着爬虫已经完成的工作，因为此刻其他的进程或者线程可能依然在解析网页，并且马上会加入新的URL。进程或者线程控制模块需要给报告为空的进程/线程发送临时的休眠信号来解决这类问题。线程控制模块需要不断跟踪休眠线程的数目，只有当所有的线程都休眠的时候，爬虫才可以终止。

2.2.3 Fuzzing和扫描器

Paros内嵌的Fuzzing可以根据输入数据格式通过HashMap快速从特征库中调取针对其输入的畸形注入数据，针对性很强。并且Paros支持自动监控被测目标，及时记下异常测试记录，再结合静态分析和动态调试来综合判断Web应用是否存在可利用漏洞。

2.3 Paros Proxy的优势与不足

Paros Proxy内嵌的爬虫模块实质上是一个基本的网页爬虫，它不支持对于脚本URL的解析。而实际上，目前的主流爬虫工具几乎都不支持对于JavaScript和VbScript等生成的URL的解析。但由于网络爬虫在扫描器中需要完成Web应用程序资源映射的功能，这为之后的检测模块提供资源保障，它的性能和效率直接影响着最终的扫描结果，因此爬虫模块的性能优劣对于Paros Proxy的整体工作还是很重要的。而改进并实现功能强大的网络爬虫也因此而变得十分的关键。

Paros Proxy最主要的鉴别器是它内置的漏洞扫描器。它对挖掘普通的漏洞是十分有用的，例如反射型跨站脚本漏洞、SQL注入漏洞等等。但是由于可用的数据检测包数量太少，导致Paros Proxy的扫描器不足以发现很多典型应用程序中的漏洞。并且由于Paros扫描器的检测过程只支持单线程，当你处理一个大的应用程序时，检测过程的时间将会很长。另外，由于Paros存在内置数据库，所以应用时对后期的升级维护十分不方便，而使用Paros Proxy进行漏洞扫描的优点就是它能够通过拦截代理传递修改通信请求进行攻击检测，并且它可以自动监听和记录检测结果。这个一个单独的漏洞扫描器所不具备的。

3 Paros Proxy 爬虫模块的改进

3.1 问题分析

Paros中的爬虫模块采用的是广度优先爬行策略，正如前面我们分析的，为了解决广度优先策略自身的弊端，Paros的爬行过程采用多线程进行抓取，最多支持5条线程，可最深抓取9层URL。

当前JavaScript语言被广泛使用于网页编程。但目前几乎所有的开源爬虫均没有很好的支持Javascript URLs解析，Paros的内嵌爬虫模块也同样不支持。但这样一来对动态页面中的链接就无法抓取到，使爬虫抓取链接的完整性受到较大影响。由于JavaScript在网页开发中的使用极为广泛，因此解析由JavaScript产生的URLs是十分有必要的。

3.2 解决方案

获取动态URL的难点在于脚本代码的可编程性。URL由脚本语言计算得到，脚本语言是由网页编程人员编写的，不同的网站编写的脚本千变万化。而解决此问题，通用的解决

办法只有一个，就是用脚本分析引擎去真正执行脚本代码，这样无论网页中的脚本程序多么复杂，都能得到正确结果，而本文就要用到 JavaScript 引擎。

针对此问题，本项目采用的解决方案为，引入开源 JavaScript 引擎 SpiderMonkey 去真正分析执行脚本代码。

SpiderMonkey[3] 是由开放源码浏览器 Mozilla(其前身就是著名的 Netscape Navigator) 开发小组开发的 JavaScript 引擎。SpiderMonkey 作为一个单独的组件，以 C 语言源代码形式发布，它支持 JS1.4 和 ECMAScript-262 规范。该引擎分析、编译和执行脚本，根据 JS 数据类型和对象的需要进行内存分配及释放操作。利用 SpiderMonkey 可以让应用程序具有解释 JavaScript 脚本的能力，目前已有若干个项目都采用了 SpiderMonkey 引擎，像 K-3D、WebCrossing、WebMerger 等。

3.2.1 具体实施

首先，先将 SpiderMonkey 封装，让它以一个简单的 C++ 接口呈现。写一个方法 JSParser。在这个方法里，需要先定义方法 Initialize()。在 Initialize() 中，JS 引擎收到初始化请求，执行对 JS RunTime 分配内存，而应用程序所使用的变量、对象和上下文都保存在 RunTime 中。

建立好 JavaScript 引擎后，需要在引擎的运行环境内建立网页中可能出现的 JavaScript 对象。网页中提取的 JavaScript 代码中的对象按照创建方式可以分为四类：

1）JavaScript 语言内置对象，如内置数组 (Array) 对象使得在 JS 引擎中创建和操作数组结构很容易。类似地，日期 (Date) 对象提供了一个操作日期的统一机制。内置对象使开发任务得以简化；2）JS 引擎使用的函数和全局对象。全局对象为应用程序中创建和使用的其它 JS 对象及全局变量提供缺省范围。而函数对象则使得对象具有和调用构造函数的功能；3）用户创建对象，由网页编写人员通过嵌在 HTML 语句中的 JavaScript 语句创建的对象，SpiderMonkey 引擎执行相应 JavaScript 语句便会在运行时的环境中创建；4）浏览器内置对象，这些对象由浏览器创建、维护，SpiderMonkey 引擎将其视为外部对象，必须由应用程序调用引擎的 API 创建。

第三类对象比较容易处理，本文着重介绍对于第四类对象，浏览器内置对象的初始化。

浏览器内置对象的成员函数提供了修改页面内容，设定浏览器外观甚至打开一个子窗口的途径。它通常包含十几个常用的 DOM(文档对象模型) 对象，另外还有一些使用较为灵活的浏览器内置 ActiveX 控件对象。因此本项目要做的则是模拟浏览器的工作，接收 JavaScript 语句，并设定浏览器内置对象的初始值，然后执行 JavaScript 语句中的函数。

浏览器内置对象包括 Windows 对象，Document 对象和 Location 对象。其中能处理 URL 的函数主要包括 Windows 的 open() 函数和 Document 的 write() 函数。

于是首先在 JSParser 中创建一个“基对象”，然后在 JavaScript 初始化代码中引用其 _base 属性和函数来构建浏览器内置对象。项目创建三个 _base 的成员函数，如表 1 所示。

表1 _base成员函数

名称	作用
Open()	用来实现 Windows 对象的 open 类函数
Write()	用于实现 Document 对象的 write 类函数
Href()	保存当前页面的 URL

_base.open() 主要是实现 window.open() 的功能。在程序中定义一个回调函数 window_open()，类型为 String，它的作用是将对应的 JavaScript 函数所接收的待提取的 URL 保存到本地的字符数组里。另外要将这个参数的类型由 SpiderMonkey 定义的 JavaScript 内置类型转换成 C 语言的字符数组类型。然后在 JSParser 的 Initialize() 中调用 SpiderMonkey API，将本地 C 语言函数 window_open() 映射为编写好的 _base.open() 函数。这样，以后引擎在执行 "_base.open()" 时，实际就是在回调本地的 window_open() 函数。

_base.wirte() 以 document.write() 为模板构造。它的参数是 JavaScript 内置 String 类型的 HTML 代码。document.write() 将 HTML 代码写到当前页面末尾。于是本项目的做法是，同样在程序中定义一个回调函数，document_write()，类型为 String。目的是将函数体内接收到的 HTML 代码转换成 C 语言字符数组保存到本地的 HTML 代码数组，然后再调用 HTML 代码分析子模块，从这些代码中提取 URL。为了提升网页采集准确率，尽可能多的覆盖采取，本文在这里引入递归方法，即在执行后检测输出的结果代码中是否仍含有 JavaScript，如果有，进入递归，再次调用引擎执行，以保证完整高效的解析。建立好 document_write() 函数后，与 _base.open() 一样，在 JSParser 的 Initialize() 中调用 SpiderMonkey API 进行映射，之后引擎在执行 "_base.write()" 函数时，实际上就是在回调本地的 document_write() 函数。

本文将编写好的构造浏览器内置对象的 JavaScript 代码保存为初始化文件 init.js，在 JSParser 的 Initialize() 中读取 init.js 文件并在 JSParse 内的 SpiderMonkey 引擎的运行时的环境中执行。如果需要修改浏览器内置对象，只需要修改 init.js 文件就可以。

在 JSParser 初始化后，本文定义了函数 Parse()，Parse() 的作用主要是通过 Parse()，JSParser 类对象可以调用执行页面中的 JavaScript 代码提取 URL。

当采集器分析完一个页面时，便会继续调用 JSParser 类的 getURLs() 函数。getURLs() 函数的主要作用在于，通过此函数可以获得引擎执行脚本代码后得到的 URL。要注意的是，getURLs() 函数返回的结果可能是带协议名和主机名的绝对 URL，也有可能是相对于站点根目录或者当前页面的相对 URL，这个需要由采集器从其格式判断到底是绝对 URL 还是相对 URL。

最后，需要在 JSParser 中定义 clear() 函数，clear() 函数在采集线程结束时被调用，它的主要作用是调用 SpiderMonkey

API，销毁 Initialize() 中建立的执行上下文和运行时的环境。

封装好后的 SpiderMonkey 是一个简单的 C++ 接口，于是本文将其写成动态链接库形式，在 Java 程序中 LoadLibrary，利用 native 接口，用来完成程序对封装好的脚本解析器 SpiderMonkey 的调用，以完成爬虫模块对于 JavaScript URLs 的采集。

3.2.2 实验结果

本文随机选取了一个含有大量动态网页的网站，主要测试改进后的 Paros 在页面采集功能上的正确率、召回率及覆盖率。为得到更准确的测试结果，本次试验执行五次，以获取平均数值。如表 2 所示。

表2 Paros爬虫模块改进后提取动态URL 正确率和召回率统计表

网站	动态URL	正确	遗漏	正确率	召回率	覆盖率（改进前）	覆盖率（改进后）
forum.lyd.com.cn	186	181	21	97.3%	88.8%	20	204

通过表 2 我们可以得出，改进后的 Paros 在页面采集性能上确实有很大的提升。

4 Paros Proxy 检测模块的改进

4.1 问题分析

Paros 的检测模块采用单线程方式来实现批处理方式的漏洞探测。当用户检测较大的应用程序时，把 URL 数量因素考虑进来，那么要发送的请求的数量是巨大的。另外，Paros 对于检测普通的跨站脚本漏洞是可以的，但是由于它自身的内置检测特征库，不利于软件检测特征库后期的维护及升级，另外检测数据源也不充足，因此存在漏报现象。

对于跨站脚本的检测，Paros 无法检测存在于标签属性中这类型漏洞。典型的攻击代码为：javascript:alert('xss') 。

4.2 解决方案

对于检测单线程的问题，本文采用的是引入多线程并行技术，最大限度的利用 CPU。而对于检测模块忽视上述类型的漏洞检测问题，本文也通过总结易引发上述漏洞的关键属性，编写正则表达式，在对于网页源码解析过程中，加入对于其关键字的分析检查。

4.2.1 具体实施

把一个注入过程分解成几步，让这几步的动作可以重叠。在第一步和第二步中间加入一个请求队列，这两个动作就是可以重叠的。

请求生成器不断生成新的请求放入请求队列，而发送请求的线程不断从请求队列中取出请求然后发送给服务器端。由于有请求队列的存在，请求生成和发送可以是异步的。这样就实现了请求生成和发送的重叠。

检测过程设计为一个单独的线程，对于并发检测 HTTP 请求的功能需求，本文引入线程池来满足。图 1 显示了改进后的检测模型。

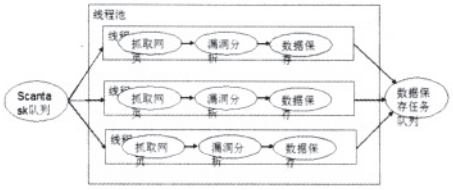

图1 改进后的检测模型

另外，对于本章提到的 Paros 忽视的那种类型漏洞的检测，本文使用了如下的检测正则表达式：

String searchExpr = ATTR + "=\\s*?([\\\"']?)" + attackString;

其中，ATTR 为易引起此类漏洞的风险属性的集合成员；attackString 为攻击字符串的正则表达式形式。通过上述检查，即可判断是否存在此类型的跨站脚本漏洞。

4.2.2 实验结果

这一章里，本文对 Paros 的检测模块进行了改进，首先是加入了线程池，使 Paros 的检测可以支持多线程，另外又对于 XSS 检测模块进行了改进，增加了一种类型的 XSS 漏洞的检测特征识别。测试时，本文主要从以下三点来考虑：1）经过改进，检测准确率明显提升；2）改进后检测速度明显加快；3）此次的模块改进不影响整体工具的功能实现。

基于以上三点，本文此次选用某政府门户网站进行测试。此次试验调用的 3 个线程，测试 5 次，以得出更精确的平均测试结果。测试结果如表 3 所示。

表3 Paros检测模块改进前后比对

	爬行深度	爬取数量	漏洞数量	耗时 /s
改进前	1	45	2	1750
改进后	1	132	6	578

而同比试验，XSS-ME 0.4.0 在检测中报错自动退出，测试失败。

5 结束语

随着网络安全问题被日益重视，Web 应用的安全漏洞测试工具被广泛用于各大商业化组织的安全测试。而选择测试工具的主要标准就在于工具的准确率及高效性等。本文通过详细的调研分析，针对目前网络安全状态面临的问题，基于一款开源工具 Paros Proxy 的综合性框架，设计并进行二次开发。改良后的 Paros 在性能及准确率上都有了明显的提高。但是，改进后的 Paros 仍存在一些缺陷，如没有考虑到跨平台问题，没有对其他漏洞特征进行深入研究等，这将会成为今后的研究方向。（责编 杨晨）

参考文献：

[1] 王清 . 0day 安全：软件漏洞分析技术 [M]. 北京：电子工业出版社，2008.

[2] 赛思教育 . Paros Proxy：网页漏洞程序评估代理 [EB/OL]. http://www.parosproxy.org/index.shtml，2011-06-07.

[3] 王映，于满泉等 . JavaScript 引擎在动态网页采集技术中的应用 [J]. 计算机应用，2004，(02)：32-36.