



一种基于匹配的 Android 系统漏洞检测方法

张嘉元

(西安电子科技大学网络与信息安全学院, 陕西 西安 710071)

摘要: 目前 Android 系统占领了智能移动操作系统的绝大多数份额, 然而 Android 系统的安全问题却令人忧虑, 近年来频频爆出高危漏洞, 给广大 Android 用户带来了很大的安全隐患。对 Android 系统漏洞检测进行了研究, 提出了一种基于匹配的快速检测 Android 系统漏洞的方法, 同时设计和实现了一个基于脚本的漏洞扫描工具。实验表明, 该工具可有效检测 Android 系统漏洞, 运行效果良好。

关键词: Android; 匹配; 漏洞检测

中图分类号: TP309.1

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2016155

A detection method of Android system vulnerabilities based on matching

ZHANG Jiayuan

School of Cyber Engineering, Xidian University, Xi'an 710071, China

Abstract: Android system has occupied the majority share of smart mobile operating systems. However, Android system has worrying security issues. It frequently bursts high-risk vulnerabilities in recent years, which is a big security risk to the majority of Android users. Android system vulnerabilities were studied, and a rapid detection method of Android system vulnerabilities based on matching was proposed. A script-based vulnerability scanning tool was also designed. Through the evaluation of experiments, the tool is running well.

Key words: Android, matching, vulnerability detection

1 引言

随着移动互联网时代的到来, 智能移动设备的普及率达到前所未有的程度, 通过移动设备上网的用户数量也超过了通过传统 PC 上网的用户数量。在这样的大环境下, Android 系统手机目前已经是世界出货量最高的手机, 用户数量也超过了其他操作系统的智能手机。据统计显示, 2013 年第 3 季度全球智能手机的销量为 2.5 亿多台, 其中, Android 系统占据了 81.9%^[1], 而截至 2014 年 11 月, 通

过 AppBrain 统计的数据可以看到, Android 官方应用市场 Google Play 上的应用数量已经接近 140 万个^[2]。

尽管 Android 设备的用户数量庞大, 但用户的安全意识却有待增强。来自美国《消费者报告》数据显示, 在美国没有采取任何安全措施的手机用户占比 40%, 2012 年有 560 万人手机遭受到账户异常访问的问题^[3]。同时由于 Android 智能手机属于开源的操作系统, 暴露出大量的安全问题, 而且由于其应用市场管理模式的原因, 其安全问题令人忧心。Android 平台的安全检测近几年虽然发展较快, 但是目

前没有一款较为全面的 Android 系统漏洞检测工具,目前手机端的安全工具基本都是对 Android 恶意应用及病毒木马的检测。这些应用都是基于对 Android 特殊的应用文件——APK 文件进行分析,判断某个 APK 应用文件是否为恶意应用,但是对于手机设备本身以及手机系统是否有安全漏洞,尚无较好的解决方案。

2013 年 8 月,Bluebox 宣布发现了一个几乎所有 Android 操作系统都存在的漏洞,并且随后公布了漏洞细节^[4]。一时间,在正常的 Android 应用中插入恶意代码,然后再进行重打包可以不被任何人察觉。对此漏洞,只有几个手机厂商进行了修复,而大量的 Android 用户暴露在危险的环境中,同时手机厂商也没有对已经出厂的移动设备发布统一的修复补丁。2014 年 7 月,Bluebox 再次发布了一个 Android 操作系统的漏洞——FakeID 漏洞^[5],并于该年的黑帽大会上公布了漏洞细节。这个漏洞同样是由 Android 系统开发人员的疏忽所导致,危害大、范围广。但是,手机厂商也无法拿出较好的解决方案检测漏洞,来保证手机用户的信息及财产安全。

综上所述,Android 用户处于较大的安全风险中,Android 系统各方面都暴露出大量的安全问题,给黑客、流氓软件、病毒、木马大量可乘之机。

出现这样的状况的原因在于 Android 系统的碎片化严重,不同手机厂商生产的系统差别很多。有的第三方系统修复了一些漏洞,而有的却又引入了新漏洞。由此导致 Android 系统参差不齐,系统漏洞难以扫描,进而也难以修复漏洞。基于以上背景,本文提出了一种基于匹配的 Android 系统漏洞的快速检测方法。

2 基于匹配的检测方法

目前安全研究人员一般将漏洞扫描方法分为主动扫描与被动扫描两种方式^[6],一般来说被动扫描效率要高于主动扫描。被动扫描一般是检查被扫描对象的安全配置及一些常见的安全弱点,例如弱密码、权限配置等问题。而主动扫描则是基于网络,通过网络协议对被扫描对象发起模拟攻击,探查对象是否存在对应的漏洞。本文所提出的基于匹配的扫描方式属于被动扫描的范畴。

所谓基于匹配的检测方法,就是匹配 Android 系统的脆弱点特征,判断待检测对象是否存在对应的漏洞。在匹配时,同时也要根据漏洞类型来确定所需要匹配的特征值。这里本文将 Android 漏洞进行重新分类,分类的主要标

准是匹配时所需的特征值提取方法。

本文对 Android 漏洞的分类如下:

- Native 代码所产生的漏洞;
- Java 代码所产生的漏洞;
- 系统应用文件相关漏洞。

这 3 类漏洞全面涵盖了 Android 常见漏洞的出现环境。本文设计并实现了一个漏洞检测工具,采用多层结构实现了 Android 漏洞的匹配式检测。

3 检测工具的设计与实现

3.1 整体框架设计

漏洞检测工具整体模块架构设计如图 1 所示,其中工具的扫描引擎输入为脚本文件,输出为扫描日志及报告,中间过程为了实现具体的扫描,需要其他模块的配合。各模块具体功能如下。

- 扫描日志及报告:用来存储扫描结果以及输出检测报告。
- 漏洞扫描辅助组件:包含很多小的模块,主要实现漏洞匹配时提供一些必要的上下文信息、收集尝试漏洞匹配后的信息、扫描过程中系统状态的上报、与扫描结果存储模块的信息交互等。
- 版本匹配器:这个模块比较复杂,主要用来匹配系统的脆弱点,从而判断系统是否存在相应漏洞。

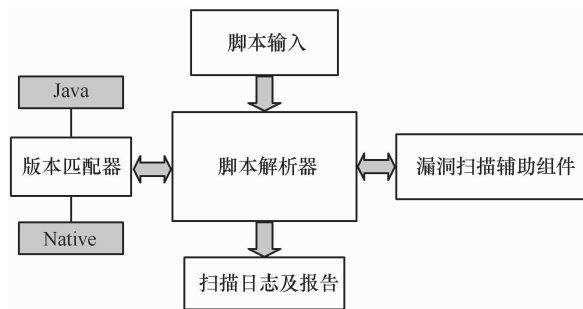


图 1 漏洞检测工具模块架构设计

3.2 脚本解析器的设计与实现

脚本语言是一类比较抽象的高级语言,它与 Java、C++ 等语言的区别在于它们是解释型语言,无须进行编译过程。编译型语言在运行之前要经过 4 个阶段,分别是词法分析、中间代码生成、优化和目标代码生成。而功能强大的脚本语言只须完成前 3 阶段的工作。目前比较流行的脚本语言,如 VBScript、JavaScript、Jython、Python、Lua、Perl、PHP 等。本文采用脚本的目的在于更快更简单地编程,更



直观清楚地编写程序逻辑,其应用特点很明显^[7]。

脚本解释器的功能主要是解释匹配脚本,同时联系各个其他模块,共同完成扫描任务。解释器状态流程如图2所示。一共分为4个状态:初始化、装载其他模块、加载脚本、运行结束。具体流程如下。

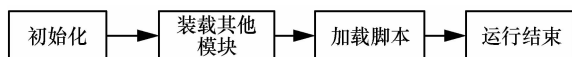


图2 脚本解释器模块流程

(1) 初始化

初始化主要是初始化一个 Lua 虚拟机。这里使用了 Luajava 开源库^[8],该开源库使用 Java 语言实现了一个 Lua 虚拟机。但是该虚拟机在使用 Lua 脚本调用 Java 代码时存在缺陷,由于 Luajava 通过反射来调用 Java 代码,而它使用 classForName 函数调用时只使用 SystemClassLoader 来查找当前类。在 Android 环境中使用的 ClassLoader 并不是 SystemClassLoader,而是定制类加载器,可以以更多方式反射更多路径下的 jar 和 class 文件。同时,去掉了脚本解释器中的冗余库文件,例如 OSLib、Bit32Lib、JmeIoLib 等,进一步减小了 Luajava 的体积。最后,将其重新封装为初始化接口 initLuajava。

(2) 装载其他模块

其他模块不论是采用 Java 语言编写还是用 C 语言编写,都会在 Java 层映射为一个独立的类,并且继承于 function.model.Base 类。这些类在 Lua 虚拟机里的装载,表现为使用 Java 反射初始化对应类的对象,然后调用对应对象的接口。装载的方式在虚拟初始化加载 Lua 脚本前初始化,通过 rawset 将对应的对象映射到 Lua 虚拟机里,也可以在 Lua 脚本里通过实现的新 Instance 接口动态加载模块。

(3) 加载扫描脚本

扫描脚本里包含定制的扫描逻辑,借助 Luajava 开源项目的 Lua 语法分析器解析 Lua 脚本,同时调用装载步骤中加载的各个功能接口或者在脚本里动态加载。

(4) 运行结束

脚本执行结束,返回状态值并释放虚拟机。

3.3 匹配器的设计与实现

匹配器主要用来对 Native 代码、Java 代码以及系统应用文件来进行匹配判断,根据它们的散列值、函数流程以及相关配置文件属性值等内容来匹配。匹配器的主要作用是进行特征值的快速提取和匹配。

- 对于 Native 层代码,由于编译后的代码均为二进制的机器码,难以从机器码中还原出原始函数流程,但是底层的函数库几乎不会变化,所以可以通过验证函数库的散列值来快速判断目标库函数里是否存在漏洞。
- 对于 Java 层代码,由于编译后的 Java 代码是字节码,所以可以从字节码中较容易地恢复出原始函数逻辑甚至函数名。所以反编译 Java 字节码,然后恢复出原始函数流程,进一步进行脆弱点的匹配。
- 对于系统应用文件,Android 应用文件里主要可能出现问题是 Manifest 配置文件。因而可以直接解包系统应用,然后读取 Manifest 文件,进而读取对应的属性值来判断是否存在漏洞。

3.4 扫描日志及报告模块设计与实现

扫描日志及报告模块是整个工具的输出模块,用来记录和展示输出结果。存储的方式主要有两种:一种是数据库方式,另一种是 XML 文本的方式。这两者的特性如下^[9,10]。

(1) 数据库的易用性

Android 本身实现了小巧的 SQLite 数据库,系统应用一般都使用内置的 SQLite。在扫描过程中可以将扫描结果方便快速地存储在数据库中。

(2) 数据库的高效率

数据库的读写速度比纯文件的读写速度快。

(3) XML 的开放性

XML 技术根据标准规范,允许在任何平台上读取和处理数据。

(4) XML 的简单性

XML 文档是纯文档,任何操作系统都可以支持文本的读写,因此可以方便地在不同操作系统中传输数据。

(5) XML 的可扩展性

XML 可以定制语言,在当前文本无法解决某些需求的情况下,可以定制例如 MathML、CML 等语言。

综合以上分析,本文选用数据库结合 XML 的存储方式。其过程为:首先创建数据库及表结构;扫描过程中将结果实时存储在 SQLite 数据库中;扫描结束后,通过数据库中的漏洞信息生成 XML 文件。这种存储方式,既保证了运行效率,同时也拥有可扩展性。扫描日志及报告模块的数据库采用系统自带轻量级 SQLite 数据库以及一个 XML 形式的报告展示。数据库的表结构见表1。同时提供访问的

接口,方便其他模块调用,接口详细信息见表 2。接口的具体实现均是以通用的 SQL 查询语句的方式。XML 形式的报告是通过数据库的信息生成,XML 的接口对象 XMLContentHandler 只有两个接口函数,分别是 genXMLFile 和 genSQLRecord。

- genXMLFile:用来通过数据库里的记录生成 XML 文件。
- genSQLRecord:用来从 XML 文件读取数据,并写回到数据库对应的表里。

表 1 扫描结果数据库结构

表名tablename)	vuldb
ID(id)	integer
漏洞名(name)	text
漏洞产生时间(time)	date
漏洞危害(detail)	text
漏洞编号(cve)	text
漏洞类型(type)	text
应用散列值(checksum)	text
应用包名(packagename)	text

表 2 模块接口信息

接口名称	主要功能
insertVul	插入新扫描记录
clearDB	清除
closeDB	关闭数据库
openDB	打开数据库
updateVulInfo	更新制定记录信息
searchVulInfo	查询记录详细信息
deleteVul	删除记录

3.5 扫描辅助组件模块设计与实现

扫描辅助组件的主要功能,是为漏洞触发器中的各个插件提供辅助作用。由于各个插件独立,需要一些通用的触发环境以及操作,本文使用扫描辅助组件实现这些通用的环境及操作,这样代码重用性更强。其中,辅助组件类型和功能如下。

- 存储接口:用来调用数据库接口,存储扫描结果。
- 状态上下文:用来存储应用的上下文以及当前扫描插件的状态。
- 信息上报接口:扫描过程中上报一些状态信息,该接口完成信息传递,并实时上报扫描状态。

扫描辅助组件主要是提供一些常用并且通用的功能,各个辅助接口都在 Lua 虚拟机初始化的时候加载进去,主

要接口如下。

- LuaConext:状态上下文接口。在调用系统接口或者启动一些其他组件时,需要系统应用的上下文 context。LuaContext 就指向当前应用的有效的 context,供其他模块在需要时调用。
- UpdateInfo:状态信息上报接口。由于系统应用中其他线程或者进程要刷新 UI 上报实时信息,需要通过 Handler 方法,在系统当前 UI 线程里定义一个处理消息的 Handler,然后进行 UI 刷新操作。UpdateInfo 接口为 UI 线程定义了一个处理消息的 Handler,并且开发了一个发送消息的接口。这样就可以在任意线程刷新 UI,实时上报信息。
- DataBaseHelper:存储接口。该接口封装了数据库的一些操作接口,同时一直处于虚拟机中,方便各模块的调用。

4 工具测试与结果评价

为了评估漏洞扫描工具,本文进行了系列的实验,验证漏洞扫描效果。使用 Android 模拟器来对检测工具与扫描方法进行测试,并且也使用其他安全工具进行了测试,最后对检测结果进行了综合性的分析和对比。

4.1 测试过程设计

由于本文提出的是一个漏洞引擎,需要配合对应的扫描脚本对系统漏洞进行检测。所以编写了一系列的扫描脚本进行辅助扫描,将编写的扫描脚本以及上层扫描策略库与本文实现的漏洞扫描引擎结合起来进行扫描测试。

4.1.1 测试方法

测试首先选取合理的测试目标,明确测试目标是对一个工具进行有效测试最重要的一个步骤。本文测试均采用 Android 模拟器进行测试,但无法对大量的 Android 系统版本进行测试,因此采用了典型与常用的系统版本进行测试,其中有 Android 2.3、Andriod 4.1、Android 4.3 共 3 个版本。测试的对象是本文实际实现的漏洞扫描引擎,同时跟目前现存的其他 Android 安全工具进行对比测试。由于事实上目前其他的 Android 安全工具并没有完善的系统漏洞检测这一功能,只有部分安全工具可以检测出 Android 系统漏洞,所以这里分别选取了 360 安全卫士、金山卫士进行了简单对比。

4.1.2 测试目的

测试目的首先在于验证本文设计的 Android 系统漏洞



扫描引擎的可行性,然后通过模拟在模拟器上运行测试,评估引擎的性能指标。本文主要记录的测试结果为扫描时间、扫描漏洞数和扫描误报数,这3个方面的结果分别体现了对应的扫描引擎的扫描效率、扫描能力和扫描准确性及可靠性。

4.2 测试结果与分析

首先比较在 Android 版本为 2.3 的模拟器中,本引擎的扫描结果与其他安全工具扫描结果的横向对比,具体见表 3。本次测试中,一个扫描脚本输入扫描引擎,然后在 Android 2.3 的模拟器里运行,获得检测结果,然后对结果进行详细分析。对扫描结果进行分析,并联系测试场景得到以下结论。

- 从扫描的漏洞数量上来看,新设计的漏洞检测工具在扫描 Android 系统漏洞上有很大优势,这是由于从设计层面上就是考虑对 Android 系统漏洞的安全检测以及环境检测。而其他安全软件主要针对的是 Android 系统上的应用扫描,包括恶意应用扫描、病毒木马扫描以及系统应用漏洞扫描。
- 从扫描的效率上看,要比其他安全工具的扫描效率高。这是由于其他安全软件的漏洞扫描主要是针对系统应用层进行,所以扫描所需时间跟系统上安装的应用数量息息相关。而本文测试使用的模拟器除了一些原生应用,并未安装其他软件,因此扫描时间很短。
- 从运行过程及扫描误报数上看,本工具跟其他安全软件一样表现良好,运行正常,并且误报数为 0。

表 3 在虚拟机 Android 2.3 中各安全软件运行结果比较

扫描项	漏洞检测工具	360 安全卫士	金山卫士
运行是否正常	正常	正常	正常
扫描时间/s	14	6	8
扫描漏洞数/个	15	2	1
扫描误报数/个	0	0	0

接下来,详细查看运行在本扫描引擎之上的各个扫描脚本的具体运行情况,这里的运行环境为 Android 2.3 模拟器,对各个脚本运行情况的纵向分析及评估见表 4。从以上结果可知,首先可以得出扫描引擎良好的可扩展性,可以基于扫描引擎编写多种多样的扫描脚本,方便地扩展新漏洞。同时从各个脚本的运行时间上来看,平均运行时间显得略长,这是基于脚本语言的执行效率的瓶颈,但是影响不大。而且不同脚本的运行时间相差较大,这是因为

不同脚本的实现方式以及检测漏洞逻辑方式各不相同。从这里可以看出,对于执行效率,也可以考虑从扫描脚本层面入手,优化扫描逻辑。

表 4 Android 模拟器中的脚本运行情况

运行项	运行情况
运行是否正常	正常
扫描脚本总数/个	50
脚本最长运行时间/ms	900
脚本最短最长运行时间/ms	10
脚本平均运行时间/ms	280

最后,本文对多个系统版本分别进行了漏洞扫描与评估,对整个漏洞检测工具进行了纵向的实验与分析,同时验证在不同版本模拟器中表现的稳定性,实验结果见表 5。

表 5 扫描工具在各个系统版本中的检测结果

扫描项	Android 2.3	Android 4.1	Android 4.3
运行是否正常	是	是	是
扫描时间/s	14	15	13
扫描漏洞/个	5	3	1
扫描误报数/个	0	0	0

从实验结果可以看出,本文设计并实现的工具在以上各个系统版本中表现正常。从扫描时间上看,扫描时间相差不多,说明扫描引擎的扫描效率除了跟系统的运行效率直接相关外,跟其他环境因素无太大关系,表现稳定。从扫描漏洞个数来看,系统各个版本的漏洞数量整体呈下降趋势,而且目前较大部分高版本的漏洞在低版本中也存在。从误报数量来看,本工具的可靠性良好,没有误报。

综上所述,本文提出的基于匹配的检测方法是可行的,实用性也很高,总体来说,可以检测到 Android 系统中的大部分漏洞,并具有良好的可扩展性。

5 结束语

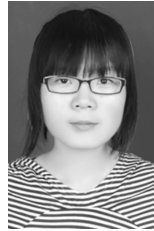
本文重点对 Android 的系统漏洞检测进行了研究与分析,从漏洞分类的角度讨论了 Android 系统中存在的威胁。然后依据对 Android 系统漏洞的分类,提出了一种基于匹配的漏洞检测方法,设计并实现了一个基于匹配的系统漏洞检测工具。经过实验测试,该工具运行良好,验证了该检测方法的可行性和有效性。

参考文献:

- [1] Gartner. World wide smartphone sales in Q3 2013 [EB/OL]. [2015-02-16]. <http://www.gartner.com/newsroom/id/2623415>.

- [2] AppBrain Stats. Number of available Android applications[EB/OL]. [2015-02-16]. <http://www.appbrain.com/stats/>.
- [3] Consumer Reports. Keep your phone safe-how to protect yourself from wireless thread[EB/OL]. [2015-02-16]. <http://www.consumerreports.org/cro/net0613.htm#info>.
- [4] WooYun. Android uncovers master-key vulnerability analysis [EB/OL]. [2015-02-16]. <http://drops.wooyun.org/papers/219>.
- [5] Blue Box. Android fake ID vulnerability [EB/OL]. [2015-02-16]. <https://bluebox.com/technical/android-fake-id-vulnerability/>.
- [6] 刘鹏. PHP Web 应用程序安全性研究及安全漏洞检测工具开发[D]. 西安:西安电子科技大学,2012.
- LIU P. Web PHP application security research and development of security vulnerability detection tools[D]. Xi'an: Xidian University, 2012
- [7] 石晓旻. 游戏引擎扩展脚本技术的研究及实现[D]. 成都:电子科技大学, 2005.
- SHI X M. Research and implementation of the game engine extension script technology[D]. Chengdu: University of Electronic Science and Technology of China, 2005.
- [8] Keplerproject. LuaJava [EB/OL]. [2015-02-16]. <http://www.keplerproject.org/luajava/>.
- [9] Wikipedia. LAMP[EB/OL]. [2015-02-16]. <http://zh.wikipedia.org/zh/LAMP>.
- [10] Wikipedia. XML[EB/OL]. [2015-02-16]. <http://zh.wikipedia.org/wiwi/XML>.

[作者简介]



张嘉元(1997-),女,西安电子科技大学在读,主要研究方向为计算机应用、网络与系统安全。