基于反编译的 Android 平台恶意代码静态分析①

李 寅, 范明钰, 王光卫

(电子科技大学 计算机科学与工程学院, 成都 611731)
(电子科技大学 信息安全研究中心, 成都 611731)

摘 要: Android 平台占有很大的市场份额, 但由于 Android 系统的开放性, 使得针对 Android 平台的恶意代码呈现出爆炸式的增长. 因此对这些恶意代码的分析和检测显得十分必要. 在传统计算机恶意代码的检测方法中, 反编译和静态分析技术占有十分重要的地位, 因此根据 Android 平台和智能手机的特点, 重点研究基于反编译的 Android 平台恶意代码静态分析方法, 并进行相关实验获取了初步的检测结果.

关键词: Android; 恶意代码; 反编译; 静态分析

Decompilation-Based Static Analysis for Malware on Android Platform

LI Yin, FAN Ming-Yu, WANG Guang-Wei

(Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)
(Research Center of Information Security, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Android platform predominates a big market share, but its openness introduces an exploding increase of malwares. So it proves quite essential to perform analysis and detection for these malwares. Static analysis plays a vital role in detecting traditional computer malwares, thus according to characteristics of Android and smart phone, this dissertation focuses on the static analysis method for Android malware based on decompilation and obtains valuable results by experiments.

Key words: Android; malware; decompilation; static analysis

1 引言

恶意代码泛指包括病毒、蠕虫、木马和后门程序等对手机操作系统会造成破坏的程序. 近几年 Android 手机的普及使得恶意代码产生了巨大的破坏力, 用户面临着巨大的经济损失和隐私泄露的威胁[1], 即通过蓝牙、红外线、GPRS 等通信技术感染手机病毒、接收到恶意程序[2]. 目前国内外在这方面的研究很少, 但黑客已开始把 Android 作为新的攻击对象, 这方面的研究已经滞后于安全形势, 成为一个亟待解决的问题[3]. 因此加强对 Android 恶意代码检测技术的研究, 具有十分重要的意义.

2 研究现状

现有的 Android 恶意代码检测方法主要使用的是计算机恶意代码的检测方法, 主要包括基于特征码、基于行为异常、以及基于神经网络的方法[4].

基于特征码的检测方法可归纳为模式的提取与匹配. 特征码是提取出的恶意代码的特征模式, 而恶意代码的扫描过程就是特征模式匹配的过程[5]. 该方法对已知恶意代码十分有效, 但其无法检测未知恶意代码, 且随病毒种类增多扫描时间会变长. 基于行为异常的检测法主要通过监控接口调用来检测, 该方法存在较高的误报率和漏报率[6]. 基于神经网络的检测方法首先需训练该网络, 并通过一定的编码规则来检测恶意代码. 该方法若增加了学习样本就需重新学习[7].

上述检测手段存在难以解决的问题. 因此本文尝试引入静态分析检测方法, 为 Android 恶意代码检测提供另外一种思路.

① 收稿时间:2012-03-29;收到修改稿时间:2012-05-07

3 基于反编译的Android平台恶意代码静态分析

3.1 基于反编译的静态分析思路

本文提出的方法的检测思路可以归纳为: 首先反编译 Manifest 文件和 DEX 文件, 将它们分别恢复为未经编译的格式; 然后归纳恶意代码中经常出现的代码序列, 设计静态分析规则.

3.2 Manifest 文件反编译

Manifest 文件包含了 APK 的配置信息. 由于本文的研究重点是探索 DEX 文件的反编译及静态分析方法设计, 因此 Manifest 文件的反编译将借助 AXMLPrinter2 工具完成, 在此不再赘述.

3.3 DEX 文件反编译

3.3.1 DEX 文件格式分析

根据 Michael Pavone 发布在 http://www.retrodev.com/android/dexformat.html 的分析报告[8], DEX 文件格式包括以下几个部分:

(1) File Header, 包含 APK 的校验和以及指向其他结构体的绝对偏移量, 包括文件大小、类列表、域列表以及方法列表等.

(2) Method Table, 包含 DEX 文件中所有类方法所属的类名、方法名和方法类型描述符.

(3) Method List, 包含特定类的方法的访问权限及代码实现的绝对偏移量.

(4) Code Header, 包含实现特定方法的代码信息, 包括局部变量、实际代码和异常等的绝对偏移量, 该偏移量指向一个 32 位整型, 在整型后紧随真实的虚拟机指令.

3.3.2 DEX 文件反编译实现

反编译的重点在于使用 Method Table, Method List 和 Code Header 等将编译后的文件还原. 我们提出以 File Header 的起始地址为基准, 根据绝对偏移量定位上述各表的起始地址, 然后再叠加表内偏移量, 获取代码中定义的方法的访问权限、接口、域和 API 调用等信息, 具体实现方法主要分以下四步进行.

(1) 实例化 EncodedMethod 数组.

(2) 初始化 EncodedMethod 数组. 包括按照 unsignedLeb128 格式设置数组长度及元素, 以上一个 EncodedMethod 对象为基准, 以叠加 Method List 等表内偏移量的方式迭代实例化 directMethods 和 virtualMethods 对象, 从 DEX 文件读取信息.

(3) 计算 EncodedMethod 数组元素大小. 在每次迭代中, 由上一个 EncodedMethods 对象和相对偏移量计算 directMethods 和 virtualMethods 数组元素的大小, 为下一步输出做准备.

(4) 输出信息. 读取 directMethods 和 virtualMethods 数组内容, 根据上一步计算的数组元素大小, 按 Dalvik opcodes 格式循环输出信息.

按照以上方法, 本文反编译了 DEX 文件, 并以“兔年日历 iCalendar”程序为例进行了测试, 图 1 是反编译后的 DEX 文件.

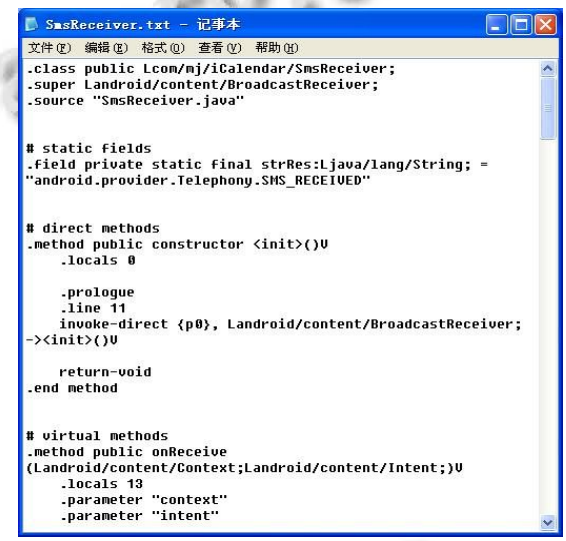

图 1 DEX 文件反编译结果

3.4 静态分析规则设计

要设计静态分析规则, 必须先统计恶意代码常用的权限和 API, 因此我们根据部分样本进行了统计, 如表 1 和表 2 所示, 其中风险等级随数字减少依次降低.

表 1 Android 恶意代码常用权限

权限	风险等级	出现频率
CALL_PHONE	5	0.25
INSTALL_PACKAGES	4	0.25
INTERNET	2	0.875
PROCESS_OUTGOING_CALLS	4	0.125
READ_CONTACTS	3	0.125
READ_PHONE_STATE	3	0.75
READ_SMS	4	0.125
RECEIVE_MMS	2	0.125
RECEIVE_SMS	4	0.25

表 2 Android 恶意代码常用 API

API	风险等级	出现频率
abortBroadcast()	1	0.25
getDeviceId()	5	0.75
getInstalledPackages()	3	0.25
getSimSerialNumber()	5	0.375
getSimState()	5	0.125
getSubscriberId()	5	0.25
sendTextMessage()	5	0.375
abortBroadcast()	1	0.25
getDeviceId()	5	0.75
getInstalledPackages()	3	0.25

根据上述统计数据，我们设计了如下的静态分析规则:

(1) 根据常用权限列表扫描反编译后的 XML 文件，确定匹配项，然后由匹配条数、各项风险等级和对应出现频率，计算加权平均值 Ax(保留两位小数，理论上限值为 5.00).

(2) 类似第一步，根据常用 API 列表扫描反编译后的 Dalvik opcodes，同样确定匹配项，然后由匹配条数、各项风险等级和对应出现频率，计算加权平均值 Aa.

(3) 最后由上述两个加权平均值 Ax 和 Aa 求平均值，四舍五入保留两位小数后作为最终风险等级值，数值越大表明风险等级越高，反之则风险等级较低.

3.5 实验结果分析

由上述执行流程和静态分析规则，我们实现了基于反编译的 Android 平台恶意代码静态分析工具并进行了测试. 测试表明，恶意代码的平均风险等级值在 3.0 以上，而正常程序的风险等级值在 3.0 以下. 因此我们将该值作为阈值，并使用该工具和几种常用杀毒软件对获取到的 Android 恶意代码样本进行了对比检测. 如表 3 所示，其中 √ 表示该样本为恶意代码.

测试表明，该静态分析工具的检测准确率达到了 82.6%，能较有效地检测出恶意代码. 但同时，在上表所示的 8 款恶意程序中存在 2 款漏报，检测准确率还有待提高.

4 结语

本文的创新点在于将反编译和静态分析引入到 Android 恶意代码检测，并提出了基于反编译的静态分析检测工具. 经测试，该工具能较有效地检测 Android 恶意代码，但目前检测准确率不是很高，这是由于恶意代码较少且使用的检测方法较单一造成的. 因此后续工作应侧重于收集更多样本，并结合使用动态检测等方法，以提高检测准确率.

表 3 部分 Android 恶意代码对比检测结果

测试样本	静态分析工具	网秦	360	McAfee
兔年日历iCalendar	√	√		√
天天美图		√	√	√
Balloon Game	√	√		√
Elite Force	√	√	√	
Love Heart Wallpaper	√	√	√	√
Stripper Touch Girl	√		√	√
Trojan-Spy.AndroidOS.Spitmo.a	√	√	√	√
Wild Man		√	√	√

参考文献

1 潘娟.移动终端的信息安全.2009 信息通信网技术业务发展研讨会.2009: 94–99.

2 左强,郝玉洁,刘乃琦.基于 Symbian 的智能手机防火墙研究与设计.微计算机信息,2008,3–3:39.

3 宋杰,党李成,郭振.Android OS 手机平台的安全机制分析和应用研究.计算机技术与发展,2006(6):153–154.

4 孙萌,落红卫.移动终端安全威胁和防护措施.现代电信科技,2009,(11):23–25.

5 金庆,吴国新,李丹.反病毒引擎及特征码自动提取算法的研究.计算机工程与设计,2007,28(24):5863–5866.

6 陈雅娴,袁津生,郭敏哲.基于行为异常的 Symbian 蠕虫病毒检测方法.计算机系统应用,2008,17(11):49–51.

7 梅华威,张铭泉.基于 BP 神经网络的手机病毒检测方法.计算机应用与软件,2010,27(7):283–284.

8 Pavone M. Dex File Format. http://www.retrodev.com/android/dexformat.html. 2011,12.