学　　号　2015301500150
密　　级

武汉大学本科毕业论文

移动应用的动态行为捕获

院 (系) 名 称:　国家网络安全学院

专 业 名 称:　信息安全

学 生 姓 名:　蹇奇芮

指 导 教 师:　傅建明　教授

二〇一九年五月

BACHELOR'S DEGREE THESIS
OF WUHAN UNIVERSITY

Dynamic behavior capture for mobile applications

School (Department): School of Cyber Science and Engineering

Major: Information Security

Candidate: QiRui Jian

Supervisor: Prof. JianMing Fu

Wuhan University

May, 2019

郑 重 声 明

本人呈交的学位论文, 是在导师的指导下, 独立进行研究工作所取得的成果, 所有数据、图片资料真实可靠. 尽我所知, 除文中已经注明引用的内容外, 本学位论文的研究成果不包含他人享有著作权的内容. 对本论文所涉及的研究工作做出贡献的其他个人和集体, 均已在文中以明确的方式标明. 本学位论文的知识产权归属于培养单位.

本人签名: _____ 日期: _____

摘　　要

智能移动终端设备的盛行使得针对移动操作系统的恶意应用迅速增加. 目前的移动操作系统中, 安卓系统巨大的市场份额和其相对开放的应用分发和权限管理方式使得其成为的攻击者的主要目标. 为了识别出恶意应用并阻止其传播, 我们需要对应用的行为进行分析. 然而单纯的静态分析在如今安卓应用成熟的混淆和加壳机制的保护下无法很好的揭示应用的行为, 因此需要动态的对安卓应用的行为进行捕获和分析.

本文分析了目前已有的一些安卓系统应用行为监测系统的实现方式和优缺点, 并且通过 hook 技术以及对安卓 8.1 源代码的修改设计和实现了一个运行于 Nexus 5x(Google 的一款智能手机) 的高性能应用动态行为捕获系统. 该系统能够捕获到 Java 层的所有方法调用以及 Native 层的重要函数调用, 并且支持动态地调整需要监控的目标方法 (Java 层) 和函数 (Native 层). 本文使用常用应用对该系统进行了测试, 结果显示与同样能捕获到所有 java 层方法调用的 Android Device Monitor 相比本系统的性能开销明显更低.

本文设计思路结合了 hook 技术带来的灵活性和以及修改源代码的稳定性以及高性能, 对其他开源平台的类似工具设计有一定参考作用, 但在应用中应当注意两种方式可能的冲突问题.

关键词: 安卓应用; 动态行为; 高性能

ABSTRACT

Malicious applications on mobile operating systems boom with the prevalence of smart mobile devices. The huge market share of Android, one of current mobile operation systems, and its relatively open application distribution and privilege management make it attackers' major target. In order to identify malicious applications and prevent them from spreading, we need to analyze the behavior of applications. However, only static analysis can not handle the mature obfuscation and packing mechanism of Android applications, so it is necessary to dynamically capture and analyze the behavior of Android apps.

In this paper, I analyze the implementation, advantages and disadvantages of some existing Android application behavior monitoring systems, and present a high-performance application dynamic behavior capture system, which can run on Nexus 5x and is implemented by using hook technology and modifying Android source code. The system captures all method invocations of Java layer and important function calls of Native layer, and supports dynamic adjustment of the target methods (Java layer) and functions (Native layer) that need monitoring. I evaluate the system with common applications and the result shows that the overhead is significantly lower than that of Android Device Monitor when capturing all java layer method invocations.

The design of the system in this paper combines the flexibility brought by hook technology and the stability and high performance brought by modification of source code, which can be used for designing similar tools of other open source platforms, but we

should pay attention to the possible conflicts between the two approaches;

Key words: Android application; dynamic behavior; high performance

目　　录

1 绪论

1.1 研究背景与意义

随着移动互联网和物联网的蓬勃发展, 智能移动终端设备迅速普及。截止 2018 年 9 月, 国内智能手机用户数量已达到 7.8 亿[5], 占总人口数的 55% 以上。如此众多的用户极大地促进了移动应用的发展, 2018 年的数据显示[7], 谷歌公司的 Google Play 上已经有超 260 万应用软件, 苹果公司的 App Store 上也有超过 200 万应用软件可供下载使用。这些应用软件涵盖了娱乐, 社交, 购物, 出行, 金融服务, 身份服务等等领域, 极大地便利了人们的生活。但与此同时, 各种服务通过应用软件集中于智能手机使得智能手机与个人隐私, 财产安全甚至人身安全的联系变得更加紧密, 从而不可避免地吸引了大量攻击者开发和传播恶意应用来牟取不正当利用。

目前市场上的智能移动终端设备运行的移动操作系统几乎均为 Android 和 Apple iOS[6]。其中 Android 以其免费, 开源的特点吸引了大量智能手机厂商, 占据了超过 75% 的市场份额[6]. 最新数据显示, 在 2019 年 Q1 国内智能手机销售量中搭载 Android 的手机销量占比达 78.2%[3]。然而, Android 本身宽松的权限管理以及开放的应用分发方式使其很容易受到攻击, 加上巨大的市场体量, 造成了绝大多数恶意应用把 Android 作为攻击目标的局面。虽然近年来 Android 的权限管理和安全机制不断加强, 同时工信部各大应用分发平台的监管加强, 一定程度上遏制了恶意应用的发展, 但数据显示[9]2018 年 Android 新增恶意软件达 800.62 万个, 感染用户数近 1.13 亿, 数量仍然庞大。另外, 恶意软件的类型也持续朝着多样化隐秘化方向发展，新式的恶意软件通过更加难以分析的加壳和混淆技术隐藏自己的恶意行为, 绕过安全软件的查杀。因此, Android 平台的安全问题依然严峻。

为了阻止恶意应用被下载运行, 保护用户手机的信息安全, 各大应用分发平台需要能够精确有效地判断开发者提交的应用是否为恶意应用, 而捕获应用的行为是分析一个新应用是否为恶意应用的必要前提。对应用软件的行为获取方法有两

个大类: 静态方式和动态方式。静态方式即在不运行应用软件的情况下对应用软件内部的资源文件、代码、数据等进行分析, 获取应用的特征, 代码逻辑等; 动态方式则是运行应用软件, 在执行过程中对软件的代码执行路径, 数据访问等进行监控和记录。在目前 Android 平台的应用加壳和混淆技术成熟的情况下, 单独的静态分析无法获取包含应用的真正逻辑的代码和数据, 因而无法获取到应用行为, 必须通过动态的方式才能捕获到包含应用真实目的的行为, 获取相应的数据, 从而判断应用是否为恶意应用。另外, 动态分析还能够在运行中捕获到执行应用真正逻辑的代码和数据 (脱壳), 从而结合静态分析揭示更加完整的应用行为。因此, 对 Android 平台移动应用的动态行为捕获技术进行研究, 有助于识别和分析隐蔽性越来越强的恶意应用, 从而遏制恶意应用的传播, 提升 Android 平台的安全性。

1.2 国内外研究现状和发展方向

Android 系统从发布至今已有 10 年, 目前国内外已有许多 Android 应用动态分析相关的研究成果发表。这些成果借鉴了传统 PC 平台的动态分析方法, 并结合了 Android 平台的自身特点, 在本地指令层面, 系统调用层面, 本地函数层面, Java 指令层面, Java 方法层面中部分或全部层面对应用的运行进行跟踪记录, 并在此基础上结合污点传播技术实现了隐私数据泄露的检测功能, 结合对应用加壳混淆机制的研究实现了脱壳和去混淆功能, 给恶意应用的分析提供了许多强有力的工具。下文介绍了一些有代表性的成果。

Enck William 等构建了名为 Taintdroid[2] 的隐私数据跟踪系统。该系统采用了污点传播技术, 通过修改 Android 系统 Java 层与隐私数据获取相关的 API 给隐私数据添加标记, 通过修改 Android Runtime 的 Dalvik 虚拟机运行机制实现了带标记隐私数据在虚拟机内部的透明传播, 通过修改 Android 系统进程间通信的接口实现了带标记隐私数据跨进程传播, 通过修改 Java 层文件和网络的 API 实现记录带标记的隐私数据去向, 从而能够检测到应用泄露隐私数据的行为。不过该系统有以下局限性: 1. 没有对应用的所有敏感行为进行监控, 例如拨打电话, 发送短信等。2. 没有对 Native 层的函数进行监控, 无法检测到应用通过 JNI 接口调用自身 Native 模块泄露隐私数据的行为。3. 针对特定 Android 版本, 并且不再支持 Android4.3 以后的版本使用。

Desnos Anthony 等构建了名为 Droidbox 的[1] 动态分析系统。该系统使用了 Taintdroid[2] 来监控隐私数据泄露, 另外通过修改 Android 系统源代码中敏感 API 的方式实现对 Java 层的电话, 短信, 网络, 文件, Java 类动态加载, 加密等 API 调用的监控, 能够记录应用在 Java 层的敏感行为。但该系统只涉及了 Java 层预定义的敏感 API 的监控, 没有实现对应用本地函数层调用的监控, 因此无法完整的揭示应用的行为, 另外该系统也针对特定 Android 版本开发, 并且不支持 Android4.1 之后的系统使用。

Google 公司构建了名为 Bouncer[4] 的恶意应用检测系统。该系统利用静态分析的方式识别已知的恶意软件, 同时通过动态分析方式使用虚拟机运行应用并记录其行为, 从而判断其是否为恶意应用。

Yan Lok Kwong 等构建了名为 DroidScope[8] 的动态分析系统。该系统通过修改运行 Android 系统的 qemu 虚拟机以及运行的 Android 系统来实现, 提供了对本地指令层面和 Java 指令层面的

1.3 论文主要工作

本文分析了目前已有的一些安卓系统应用行为监测系统的实现方式和优缺点, 并且通过 hook 技术以及对安卓 8.1 源代码的修改设计和实现了一个运行于 Nexus 5x(Google 的一款智能手机) 的高性能应用动态行为捕获系统. 该系统能够捕获到 Java 层的所有方法调用以及 Native 层的重要函数调用, 并且支持动态地调整需要监控的目标方法 (Java 层) 和函数 (Native 层). 本文使用常用应用对该系统进行了测试, 结果显示与同样能捕获到所有 java 层方法调用的 Android Device Monitor 相比本系统的性能开销明显更低.

1.4 论文组织结构

根据本文研究的特点, 本文的内容按如下方式组织:

第一章为绪论, 主要说明了本文课题的研究背景, 研究意义, 简述了国内外对本文课题的研究现状及发展方向, 介绍了本文的主要工作内容和文章组织结构。

第二章为背景技术介绍, 主要讲述了 Android 系统的基本架构, Android 应用的

基本结构, Android 平台的常用动态分析技术和应用保护技术, Android Runtime 的运行机制和 Frida 框架。

第三章为系统设计和实现, 详细说明了本文提出的监控系统的原理与组成结构。

第四章为实验与结果分析, 描述了实验环境, 实验方法并对实验结果进行了分析和总结。

第五章为总结与展望, 主要是整理本文所做的工作，并简要分析了本文提出系统的局限性和改进方案。

2　背景技术分析

2.1　Android 系统架构

Android 是由 Google 开发的一种移动操作系统, 该系统基于修改后的 Linux 内核构建, 由 5 层结构的软件栈构成, 如图2.1所示。

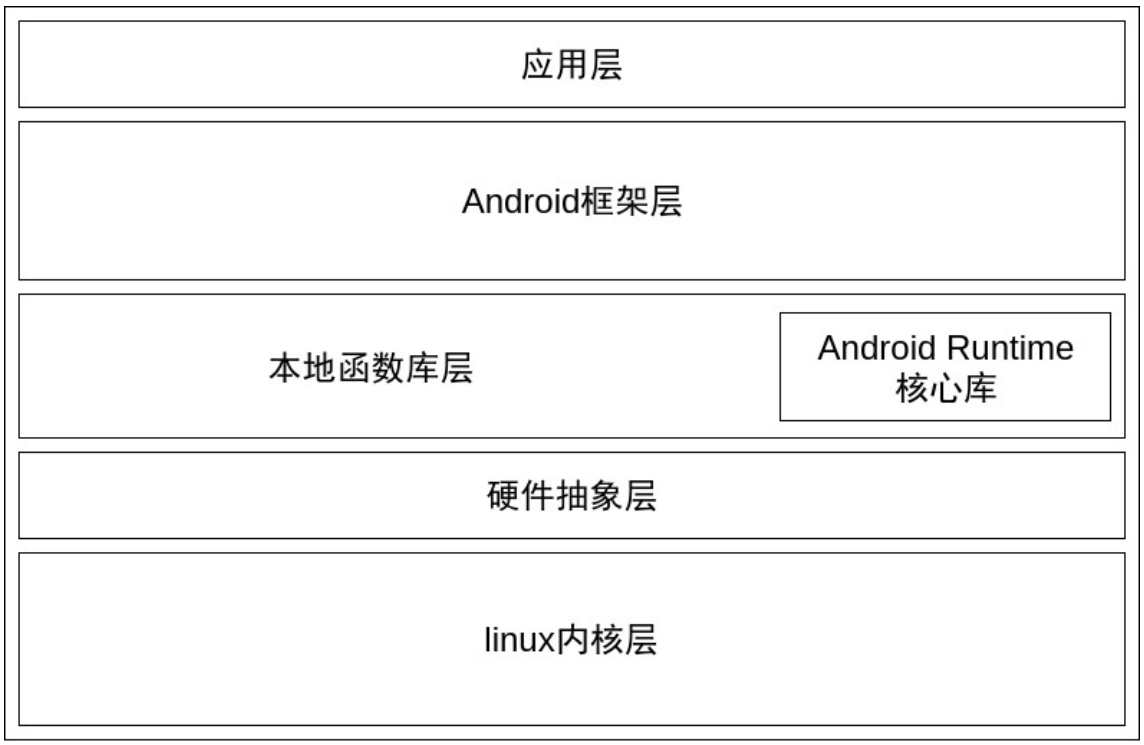

图 2.1　Android 系统架构

2.2 Android 应用结构

2.3 Android 动态分析技术

2.4 Android 加壳和混淆技术

2.5 Android Runtime

2.5.1 简介

2.5.2 应用的启动

2.5.3 应用的加载

2.5.4 方法的执行

2.6 frida 框架

3 系统设计实现

3.1 概览

3.2 启动监控

3.3 Java 方法执行监控

3.4 native 函数调用监控

3.5 log 系统

4　实验与结果分析

4.1　监控数据

4.2　性能

5　总结与展望

本系统还不够成熟

参考文献

[1] A. Desnos and P. Lantz. Droidbox: An android application sandbox for dynamic analysis. Lund Univ., Lund, Sweden, Tech. Rep, 2011.

[2] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones [c]. In Proceedings of the 9th USENIX conference on Operating systems design and implementation. USENIX, pages 1–6, 2010.

[3] Kantar. Smartphone os sales market share evolution. https://www.kantarworldpanel.com/global/smartphone-os-market-share/, 2019.

[4] H. Lockheimer. Android and security. http://googlemobile.blogspot.com/2012/02/android-and-security.html.

[5] newzoo. Top 50 countries/markets by smartphone users and penetration. https://newzoo.com/insights/rankings/top-50-countries-by-smartphone-penetration-and-users/, 2018.

[6] statcounter. Mobile operating system market share worldwide. http://gs.statcounter.com/os-market-share/mobile/worldwide.

[7] statista. App stores - statistics & facts. https://www.statista.com/topics/1729/app-stores/.

[8] L. K. Yan and H. Yin. Droidscope: Seamlessly reconstructing the {OS} and dalvik semantic views for dynamic android malware analysis. In Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12), pages 569–584, 2012.

[9] 腾讯移动安全实验室. 腾讯移动安全实验室 2018 年手机安全报告. https://m.qq.com/security_lab/news_detail_489.html, 2018.

致　　谢