一种基于 iOS 的应用软件动态行为监测系统

王宇晓[1] 陈鑫爱[2] 章曙光[2]
[1](中国信息通信研究院 北京 100191)
[2](北京城市学院 北京 100083)

摘 要 介绍 iOS 安全机制及相关安全问题,研究分析基于 iOS 的应用软件动态行为监测系统总体架构和详细功能。提供自动化监测应用软件行为的方案,主要监测模块以添加脚本方式满足变化的业务场景和测试需求。在不同的测试场景下,使用监测系统对应用软件行为进行监测分析。

关键词 iOS iOS 应用安全 行为监测

中图分类号 TP319 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.07.048

AN iOS-BASED DYNAMIC BEHAVIOR MONITORING SYSTEM OF APPLICATION SOFTWARE

Wang Yuxiao[1] Chen Xin'ai[2] Zhang Shuguang[2]
[1](*China Academy of Information and Communications Technology*, *Beijing 100191*, *China*)
[2](*Beijing City University*, *Beijing 100083*, *China*)

Abstract This paper introduces iOS security mechanisms and related security issues. It studies and analyzes the overall architecture and detailed functions of iOS-based application software dynamic behavior monitoring system, and provides a solution for automatically monitoring application software behavior. The main monitoring module uses scripts to meet changing and adding test requirements for software business scenarios. In different test scenarios, the monitoring system is used to monitor and analyze the behavior of the application software.

Keywords iOS iOS application security Behavior monitoring

0 引 言

随着移动互联网的迅速发展,移动应用成为人们社交、生活的重要组成部分。Android 操作系统和 iOS 操作系统成为当前两大主流操作系统,占据巨大的市场份额。2018 年 Android 操作系统占据全球移动系统市场份额的 83%,而 iOS 操作系统占据 15%。Android 操作系统因其开源性导致应用安全漏洞、用户隐私窃取等问题层出不穷,甚至针对 Android 系统的应用软件恶意行为曾出现爆发式增长[1]。而 iOS 系统不同于 Android 开放的系统环境,在很长时间里,系统及应用软件安全问题未大规模增长和爆发,用户倾向于相信 iOS 应用安全性更高。但实际上,iOS 应用软件开发过程中经常存在因功能而忽视安全的现象[2]。另一方面,iOS 系统安全漏洞一直是业内从业人员的重点关注对象,iOS 安全研究基本都集成在对系统的安全漏洞的挖掘,iOS 应用软件安全则被相对忽视。iOS 应用经常存在程序包可被篡改、不安全数据传输、敏感数据任意存储等问题。

Arxan 在《第 5 次应用安全年度现状安全报告》中明确提及:虽然 iOS 操作系统通常被认为相比 Android 更安全,但在研究中,iOS App 却比 Android 平台有着更多漏洞[3]。iOS 应用在运行过程中需向用户询问申请访问隐私如照片、通讯录、定位等,用户同意之后,应用访问、上传用户隐私内容均可在用户无感知的情况下发生。访问或上传用户隐私是否符合用户意向,是否存在超出用户意志行为,都应根据应用软件实际动

收稿日期:2019-05-06。国家科技重大专项(2017ZX03001018);北京城市学院 2018 年度实培计划项目。**王宇晓**,工程师,主研领域:移动安全。**陈鑫爱**,本科生。**章曙光**,副教授。

态行为判断。

目前很多基于 iOS 的应用安全检测工具和方案大多是离线型,几乎没有对 iOS 应用实时检测的工具。因此本文基于 iOS 应用安全机制,构建 iOS 应用软件动态行为监测系统,从系统底层对应用软件动态行为实时监测并输出,包括调用系统拨打电话、调用短信、网络数据传输等多方面,以达到可实时获取并分析应用软件恶意行为的目的。

1　iOS 相关安全机制和检测框架

1.1　iOS 安全机制

iOS 操作系统从硬件和固件到应用软件,设计一套安全保护机制,如安全启动链、Apple 根证书、程序代码签名、文件数据保护、沙盒技术等,以保护系统和应用的安全性[4]。尽管存在详细的安全机制,但仍存在可被利用的安全问题。

iOS 操作系统通过 App Store 作为唯一渠道,提供应用软件下载、安装和使用。当开发者将应用软件上传至 App Store 时,App Store 会对应用软件进行证书签名处理,App Store 使用 Apple 官方维护的私钥对 App 进行签名。当用户下载、安装应用时,iOS 系统会使用公钥验证此 App 是否来自官方。数字签名证书对可执行文件编码进行改变以达到保护的目的。因而,业内破解 iOS 应用时首先需要对 App Store 来源的应用软件壳进行破解才可获取真实可执行文件,此种行为通常被称为“砸壳”,常见脱壳工具有 Clutch[5]、dumpdecrypted[6]等。国内一些 iOS 第三方应用商店如 PP 助手、爱思助手等提供砸壳后的应用程序。同时苹果公司提供企业开发者证书和内测版开发者证书,可通过这两种证书对应用软件签名,签名的应用无法提交到 App Store,但是可作为内部测试使用,安装到 iOS 操作系统后只需信任应用描述文件即可使用。目前国内蒲公英分发平台提供企业签名服务。企业开发者证书和内测开发者证书的存在和使用为在非越狱设备上监测应用动态行为提供可能性。

iOS 系统“越狱”一直是业内安全研究人员的重点研究内容,其利用 iOS 设备硬件和软件漏洞,突破 iOS 系统限制。原始 iOS 系统的 Root 权限对特定私有进程外的其他进程不开放,使用 Root 权限运行的进程则可任意读取设备的文件系统,越狱后,用户可对系统进行编辑或是运行不被苹果公司所验证的软件[7]。越狱后的设备,用户可根据自己的需求,定制化更改系统。在越狱设备上对应用软件进行动态行为具备更高的便捷性,但仍存在实际问题。iOS 系统通常会对可被利用的漏洞在新更新的版本上进行修补,因而可被越狱的设备系统版本相对固定,迭代较慢。针对越狱设备的应用行为监测系统则无法在最新的 iOS 系统版本上使用,存在一定的局限性。

1.2　Frida 检测框架

Frida[8]是一款基于 Python 和 JavaScript 的 Hook 调试框架。允许将 JavaScript 的部分代码或者自定义库注入 Windows、Macos、Linux、iOS、Android 以及 QNX 的原生应用中,同时能完全访问系统的内存和功能。根据实际测试经验,Frida 框架可提供 Hook 系统 API 并修改返回值、嗅探网络数据、提取应用软件类和类方法信息。Frida 作为一款跨平台的框架,安全测试优势在于此框架可为非越狱设备上使用的应用软件插入 FridaGadget. dylib,展开基于非越狱设备的应用监测。现有几种基于 Frida 框架的安全分析工具,如 Needle[9]和 AppMon[10],可以为应用软件提供安全评估。

2　基于 iOS 的应用软件动态行为监测系统设计

2.1　设计思路

正常模式下,应用软件调用系统 API 行为指令直接发送到 iOS 系统层面。iOS 应用动态行为监测系统在应用和 iOS 系统中间层添加监测模块,监测模块监控行为、记录行为及发生时间等具体信息后返回服务器端,如图 1 所示。

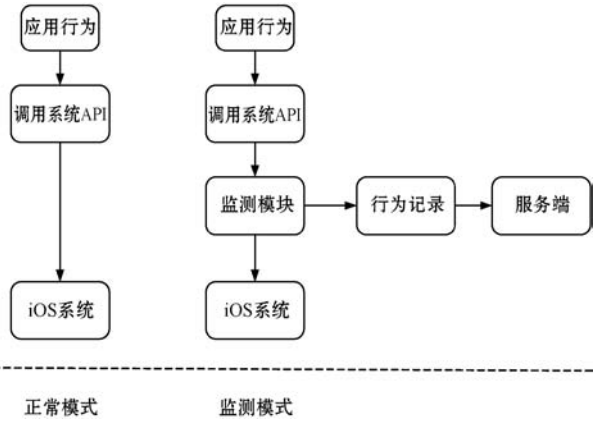

图 1　iOS 应用软件动态行为监测系统总体设计

iOS 应用动态行为监测系统设计思路是利用框架编写脚本,对应用软件进行 Hook,记录其调用系统 API 的发生时机和发生的上下文环境,以分析其是否存在超出用户意志收集、滥用用户隐私及数据的行为。本系统既提供在非越狱设备上使用企业开发者或内测开发重签名的应用软件动态行为监控,也提供越狱设备

应用软件动态行为监测。同时本系统方案提供自定义行为监测脚本配置,可由用户自主选择多种行为结合的监控方案以达到定制化监测的目的。

2.2 总体架构

基于 iOS 的应用软件动态行为监测系统总体架构如图 2 所示。系统架构主要分为两部分,分别运行于被测电脑和被测设备。总体架构中,用户端和服务端运行于被测电脑,用户端可配置基础监测信息如设备是否越狱,以及配置监测行为脚本。服务端根据基础配置信息和监测行为脚本,自动化执行对应用打补丁、重签名打包及注入监测模块等操作对应用行为监测。被测设备正常打开运行被测应用软件,应用调用系统 API。

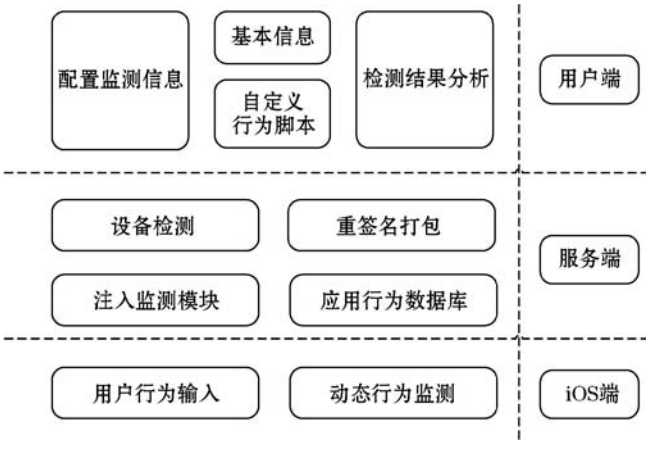

图 2 iOS 应用软件动态行为监测系统总体架构

3 系统功能

3.1 监测信息基本配置功能

监测信息基本配置功能,因基于 iOS 的应用软件动态行为监测系统面向对象包括非越狱设备和越狱设备,所以在使用系统时需根据设备不同状态进行基本配置。

(1) 非越狱设备。在非越狱设备上监控应用软件动态行为时,需先使用脱壳类工具如 Clutch 对 App Store 下载的 ipa 应用文件脱壳,或者直接使用国内第三方应用商店如 PP 助手等提供的 ipa 文件。监测系统提供应用软件企业或内测开发者证书配置功能,利用 Frida 框架将监测模块注入脱壳后的 ipa 文件,利用开发者证书重新打包,安装回非越狱设备后即可监控。如图 3 所示。

图 3 非越狱设备应用配置过程

(2) 越狱设备。在越狱设备上监控应用软件动态行为时,越狱状态下,用户掌握对系统的控制权,可随意根据需求更改系统。用户只需在 Cydia 中安装 Frida 终端设备框架,即可与监测系统进行通信。

3.2 应用软件行为监测功能

iOS 应用软件安全形势严峻,根据 iOS 应用软件安全评估实践和 iOS 常见安全问题,发现存在用户隐私信息泄漏、用户未知情况下执行恶意操作、篡改内存数据、修改系统文件内容、频繁定位等问题。针对以上内容,应用软件行为监控功能模块,梳理总结相关模块,已经实现进程间通信、网络通信、数据存储、加密功能、日志记录、Key Chain 访问、Https 数据监控、通讯录读取监控、短信发送、电话拨打、读写 User 设置监控等多个脚本模块,脚本模块和函数名示例见表 1。用户可在服务端配置监控脚本,可新增被监控的模块和函数名,监控脚本覆盖 C 类型和 OBJC 类型。

表 1 常见应用调用系统 API 示例

模块	函数名
Crypto	SecCertificateCreateWithData
Crypto	CC_MD5
UserDefaults	+[NSUserDefaults standardUserDefaults]
HTTP	NSURLSession
Contacts	ABAddressBookCreateWithOptions
Identifiers	-[UIDevice identifierForVendor]

3.3 原型系统实现

原型系统中系统用户端和服务端使用 Python 语言开发,便于在搭载不同操作系统的测试电脑上移植使用。应用动态行为监测脚本根据 iOS 系统 API 进行开发实现,脚本采用 JavaScript 语言开发,开发人员根据新业务要求新增行为监测脚本开发,无需编译可即时使用。

3.4 监测结果输出与分析

将设备接入基于 iOS 的应用软件动态行为监测系统后,完成基本配置和脚本配置。测试人员对应用软件操作分为正常使用应用软件和静置应用软件,系统监控应用软件行为并输出调用时间、调用函数、调用模块和具体细节。测试人员可根据测试需求和测试场景,分析测试结果。

原型系统中,对一款外卖类应用软件进行监测。测试场景包括正常运行使用场景和静置场景。监测系统安装环境为 Windows 7,被测手机设备为 iPhone 6S,系统版本为 10.1.1。测试结果如下:

(1) 正常场景。正常使用场景下,应用软件在正常使用过程中,读取通讯录、网络连接通信等调用系统 API 行为都是在用户业务场景触发后发生,不存在用户未知情况下执行窃取用户隐私的恶意行为。

(2) 静置场景。静置场景下,应用程序静置 1 小时,前台无任何操作。行为监控输出结果显示,应用程序多次调用 UserDefaults 模块查询用户偏好设置,其中包括 GPS 定位数据。同时网络传输数据行为显示,1 小时内应用每隔 10 分钟向固定服务器发送定位信息。在静置状态下,固定时间和频次传输定位信息的行为疑似应用程序在主动收集用户位置迁徙信息,存在用户信息泄漏风险。

4 结　语

因 iOS 操作系统闭源和封闭的特性,用户对于 iOS 操作系统的安全性认可度较高,普遍认为 iOS 应用软件安全性高。但从现实角度来看,搭载于 iOS 操作系统的应用软件安全性并不比 Android 应用软件安全性高。随着互联网的飞速发展,安全和用户隐私保护已经成为用户在使用移动智能设备过程中的迫切诉求。本文介绍的基于 iOS 的应用软件动态行为监控系统可实时对 iOS 设备上的应用软件动态行为进行监测,覆盖范围包括非越狱设备和越狱设备。应用软件实时动态行为分析是应用安全性分析的重要组成部分。应用软件实时动态行为监控可避免应用软件开发者在用户未知情况下,随意访问、窃取用户隐私信息,给用户带来经济和财产的损失。但本系统中,对非越狱设备上应用软件监控,需对源自 App Store 的应用软件执行脱壳、注入监控模块、重打包、重签名等一系列操作。随着操作系统本身安全性增强以及开发者安全开发意识的提高,应用程序可采用多种手段进行加固,脱壳难度也会随之增高。因此,本系统对非越狱设备监测部分,可能会随着应用安全性的提高而产生局限性和适配性的问题。针对越狱设备监测应用部分,无需对应用软件进行改动,只需保证将一台越狱设备接入系统即可。越狱设备的操作系统版本必然低于当前最新稳定运行版本,可能存在高版本的安全功能向下不适用的问题。因此 iOS 应用软件动态行为监测仍然留有研究改进空间。应用软件安全监测技术的进步是移动应用产业良性发展的重要保障,对促进应用软件安全监测平台的发展具有重要意义。

参 考 文 献

[1] 朱易翔,张慷,王渭清. iOS 恶意应用分析综述[J]. 电信科学,2017,33(2):42 - 47.

[2] 舒远仲,王娟,梁涛,等. 一种 iOS APP 安全评估方案[J]. 网络安全技术与应用,2018(1):11 - 13.

[3] Arxan. 5th Annual State of Application Security Report[R/OL]. https://www. arxan. com/wp-content/uploads/2016/01/State_of_Application_Security_2016_Consolidated_Report. pdf.

[4] Apple Inc. iOS 安全保护[EB/OL]. https://www. apple. com/cn/business/site/docs/iOS_Security_Guide. pdf.

[5] Kim Jong-Cracks. KJCracks/Clutch: Fast iOS executable dumper[OL]. [2017 - 07 - 06]. https://github. com/KJCracks/Clutch.

[6] Renard M. Practical ios apps hacking[C]//GreHack 2012, Grenoble, France,2012:14 - 26.

[7] Keller M. Geek 101: What Is Jailbreaking? [J/OL]. PC World, 2012, 30(5). https://www. pcworld. com/article/249091/geek_101_what_is_jailbreaking_. html.

[8] Ravnås O A V. The Engineering Behind the Reverse Engineering [OL]. 2015. http://www. frida. re/docs/presentations/osdc-2015-the-engineering-behind-the-reverse-engineering. pdf.

[9] 徐小天,陈乐然,孙跃,等. 移动应用安全检测技术研究[C]//2017 智能电网发展研讨会论文集. 2017.

[10] Patnaik N D. Appmon: runtime security testing & profiling framework for native apps[OL]. [2017 - 07 - 06]. https://github. com/dpnishant/appmon/wiki/2. -Introduction.

(上接第 220 页)

[11] 顾益军,夏天. 融合 LDA 与 TextRank 的关键词抽取研究[J]. 现代图书情报技术,2014,30(7):41 - 47.

[12] Wen Y, Yuan H, Zhang P. Research on keyword extraction based on Word2Vec weighted TextRank[C]//2016 2nd IEEE International Conference on Computer and Communications(ICCC). IEEE,2016:2109 - 2113.

[13] Qiu Q J, Xie Z, Wu L, et al. Geoscience keyphrase extraction algorithm using enhanced word embedding[J]. Expert Systems with Applications,2019,125(1):157 - 169.

[14] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[EB]. arXiv:1301.3781,2013.

[15] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation [J]. Journal of Machine Learning Research, 2003,3:933 - 1022.

[16] Hajjem M, Latiri C. Combining IR and LDA topic modeling for filtering microblogs[J]. Procedia Computer Science, 2017, 112:761 - 770.

[17] 刘冰玉,王翠荣,王聪,等. 基于引力因子的加权网络重叠社区识别算法[J]. 计算机科学,2016,43(12):153 - 157.