北京邮电大学

硕士研究生学位论文开题报告

学 号: 2015110724

姓 名: 贾晓冬

学 院: 计算机学院

专业(领域): 智能科学与技术

研究方向:

导师姓名: 梁洪亮

攻 读 学 位: 工学硕士

2016年11月25日

|  |  |  |  |
| --- | --- | --- | --- |
| 论文题目 | 安卓系统隐私泄露检测及防御 | | |
| 选题来源 | 国家自然科学基金项目 | 论文类型 | 基础研究 |
| 开题日期 | 2016-11-19 | 开题地点 | 科研楼809 |
| **一、立题依据（包括研究目的、意义、国内外研究现状和发展趋势，需结合科学研究发展趋势来论述科学意义；或结合国民经济和社会发展中迫切需要解决的关键科技问题来论述其应用前景。附主要参考文献目录）（不少于800字）**  1.1研究的目的和意义  Android是一种基于Linux的自由及开放源代码的操作系统，由Google公司和开放手机联盟领导及开发。由于其开放的特质，吸引了一大批硬件厂商和软件开发者。第三方的统计数据显示，2016年Android占有的市场份额高达76.4%，远远超过其他智能手机厂商。据不完全统计，到2015年安卓市场的应用已多达百万。且安卓市场据市场研究公司Sensor Tower的最新报告显示仅在2016年第一季度，用户从iOS和Android应用市场下载的应用数量就高达172亿次。但是这些应用在给人带来巨大便利的同时，也带来很大的信息安全隐患和风险。  研究表明，乐观估计高达90%的手机应用会访问获取用户的隐私信息[1]。这意味着不仅是恶意软件，合法的手机应用也会借助各种途径获取用户的隐私信息，比如位置信息等。因此，如何有效地检测及防御安卓应用对用户个人隐私信息的窃取，已成为日益关注的热点。  以往可以有效地检测隐私信息泄露的方法分为静态分析和动态分析两种，由于静态分析方法不能检测运行时信息，例如当app开发人员使用例如JAVA反射机制，代码加密或者动态加载等技术的代码时静态分析方法就不能很好地起到检测作用。然而由于Google对安卓系统进行了大规模的改动，以往的依赖于Dalvik虚拟机的动态污点分析已经不能满足现在的需求。因此需要结合新的安卓系统的特性对动态污点技术进行改进，以实现对安卓应用窃取用户隐私信息的检测及防御。  1.2国内外研究现状和发展趋势  通常来说检测安卓app带来的用户隐私泄露的方法可以分为四类，分别是代码解析、路径捕获和分析、动态污点检测和沙盒。  对于静态分析的技术而言，很多研究人员都使用反汇编来试图精确描述运行时行为，使用程序分析技术来解析信息流。[Manuel Egele](https://www.semanticscholar.org/author/Manuel-Egele/2489809)等人针对IOS平台，使用静态分析的方法检测数据流并开发了工具PIOS[2]，从而分析得出确实有一部分app泄露了用户信息从而使得第三方可以侧写出用户的偏好和行为模式等。针对安卓平台，Lu等人提出了CHEX框架[3]，通过使用Wala计算数据流来检测组件劫持行为；AndroidLeaks[4]检测大规模的可能的隐私泄露；FlowDroid[5]可以进行相对精确的上下文、流、域、对象敏感的分析；ComDroid[6]，AmanDroid[7]，R-Droid[8],IccTA[9]和HornDroid[10]等都试图通过提升静态分析器的性能来检测安卓应用间间接的数据流。DroidSafe[11]使用一个精确的分析插桩技术来模型化运行时，在静态代码中捕捉易忽略的语义事件。AAPL[12]可以通过多重特定静态分析技术来检测隐私泄露，并且通过一个新型的同类投票技术来进一步精准化结果。AppAudit[13]将静态分析和动态分析相结合来减少被静态污点分析引入的性能降低问题。上述的这些方法都是在app静态未执行的情况下进行分析，静态分析方法无法追踪实时的数据流和隐私泄露。  2010 年 Enck 等人在 Android2.1 操作系统上实现了动态污点跟踪系统 TaintDroid[14]， 2011 年一系列基于 TaintDroid 的系统也被提出，包括 AppFence[15]和 Mock Droid[16]等。它们都对 TaintDroid 进行了不同程度的扩展，但仍有一些缺陷并没有得到解决：①能够检测的隐私数据类型不够全面，包括了 IMEI、电话号码、地理位置信息、相册、录音，而通信录、短信息等重要信息并没有得到检测；②检测的隐私泄露点不够多，仅仅包含一般网络通信(socket)，而蓝牙、短消息等通信方式没有被监测；③字符串跟踪粒度不够细，可能导致污点跟踪过程中可能会产生污点的过度扩散而产生误报；④不能提供程序测试过程中所执行的路径信息；⑤测试时需要人工的参与，不利于自动化测试；⑥极度依赖Dalvik虚拟机进行动态污点分析。针对上述缺陷，2012年杨广亮等实现了 TaintChaser 系统[17]，对隐私信息进行细粒度的跟踪，能产生被测试程序在测试过程中所执行的路径信息，并基于TaintChaser 实现了自动化检测系统。  2014年，廖明华[18]通过针对Android应用程序常被不完全地审查，不充分的隔离，且毫无限制地被用户安装所引起的用户私有敏感信息的泄露，通过采用动态污点分析技术监测敏感信息何时通过不可信的应用程序离开系统，防止应用程序访问用户希望保密的数据。2016年Yuming Ge[1]等人对安卓app的用户隐私泄露情况作了一个深入的调研，发现90%的app都有获取用户隐私数据的行为，而这些app中的95%并不使用它们收集的这些数据，该调查也存在一定的缺陷，比如只观察了API调用导致的隐私泄露以及不能检测用户隐私信息的加密传输。  很多学者在检测用户信息窃取情况和恶意软件行为时，都用到了已被Google替换掉的Dalvik虚拟机。DroidScope[19]，BareCloud[20]，CopperDroid[21]都利用Dalvik虚拟机来获取检测恶意软件行为的动态信息；Poeplau[22]等人通过定制Dalvik虚拟机来系统地分析恶意动态代码加载；上面提到的TaintDroid以及TaintChaser等都需要用到Dalvik虚拟机。TaintDroid是一个著名的动态污点分析系统，它定制了安卓运行时即Dalvik虚拟机来实现污点存储和污点传播。很多系统都是基于TaintDroid来进行进一步分析的。但是TaintDroid已经不再适合在最近的安卓系统来进行隐私追踪和恶意软件分析，TaintDroid初始是为基于虚拟机的系统设计的，而Google已经把Dalvik虚拟机替换成了ART(Android RunTime),采用AOT编译策略，不再是在运行时使用解释器来解释代码，AOT直接在应用安装时将app编译成机器代码，TaintDroid最多可以支持到安卓4.4版本。而且越来越多的开发者倾向于开发适用于更新版本安卓系统的app，比如安卓5.0及以上版本。  2016年，针对最新的ART环境，Michael Backes等提出了ARTist[23]，实现了对ART运行时和dex2oat编译器的插桩，并且成功进行了动态污点分析，但是该系统无法对拥有多个dex文件的app进行处理，并且无法实现binder IPC和文件 级别的污点追踪。Sun Mingshen等提出了TaintART[24]，修改了后端的转换，优化和代码生成器部分，实现了动态污点分析，但是仍然存在没有对数据流进行优化，没有自动化测试系统以及只在32位处理器上实现的不足之处。  动态污点分析技术作为安卓系统中对隐私信息泄露情况进行检测常用的方法，可以有效对安卓系统中的数据流进行分析，但是安卓系统新特性的出现为动态污点分析技术带来新的挑战。本课题拟采用动态污点分析技术，对xposed框架进行定制，以实现对安卓应用的隐私信息窃取检测及防御。  参考文献  [1] Ge Y, Deng B, Sun Y, et al. A Comprehensive Investigation of User Privacy Leakage to Android Applications[C]//Computer Communication and Networks (ICCCN), 2016 25th International Conference on. IEEE, 2016: 1-6.  [2] Egele M, Kruegel C, Kirda E, et al. PiOS: Detecting Privacy Leaks in iOS Applications[C]//NDSS. 2011: 177-183.  [3] Lu L, Li Z, Wu Z, et al. Chex: statically vetting android apps for component hijacking vulnerabilities[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 229-240.  [4] Gibler C, Crussell J, Erickson J, et al. AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale[C]//International Conference on Trust and Trustworthy Computing. Springer Berlin Heidelberg, 2012: 291-307.  [5] Arzt S, Rasthofer S, Fritz C, et al. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps[J]. ACM SIGPLAN Notices, 2014, 49(6): 259-269.  [6]Chin E, Felt A, Greenwood K, et al. Poster: Analyzing Inter-Application Communication in Android[J].  [7] Wei F, Roy S, Ou X. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 1329-1341.  [8] Backes M, Bugiel S, Derr E, et al. R-Droid: Leveraging Android App Analysis with Static Slice Optimization[C]//Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. ACM, 2016: 129-140.  [9] Li L, Bartel A, Bissyandé T F, et al. Iccta: Detecting inter-component privacy leaks in android apps[C]//Proceedings of the 37th International Conference on Software Engineering-Volume 1. IEEE Press, 2015: 280-291.  [10] Calzavara S, Grishchenko I, Maffei M. HornDroid: Practical and Sound Static Analysis of Android Applications by SMT Solving[C]//2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2016: 47-62.  [11] Gordon M I, Kim D, Perkins J H, et al. Information Flow Analysis of Android Applications in DroidSafe[C]//NDSS. 2015.  [12]Mossman D, Noffsinger S G, Ash P, et al. AAPL practice guideline for the forensic psychiatric evaluation of competence to stand trial[J]. Journal of the American Academy of Psychiatry and the Law Online, 2007, 35(Supplement 4): S3-S72.  [13]Xia M, Gong L, Lyu Y, et al. Effective real-time android application auditing[C]//2015 IEEE Symposium on Security and Privacy. IEEE, 2015: 899-914.  [14]Enck W, Gilbert P, Han S, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 5.  [15]HORNYACK, P., HAN, S., JUNG, J., SCHECHTER, S., AND WETHERALL, D. These Aren’t the Droids You’re Looking For: Retrofitting Android to Protect Data from Imperious Applications. In Proceedings of ACM CCS (2011).  [16]Beresford A R, Rice A, Skehin N, et al. Mockdroid: trading privacy for application functionality on smartphones[C]//Proceedings of the 12th workshop on mobile computing systems and applications. ACM, 2011: 49-54.  [17]杨广亮, 龚晓锐, 姚刚, 等. 一个面向 Android 的隐私泄露检测系统[J]. 计算机工程, 2012, 38(23): 1-6.  [18] 廖明华. 安卓应用程序误用用户信息的监测与控制[J]. 计算机系统应用, 2014, 23(6): 215-219.  [19]Yan L K, Yin H. Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis[C]//Presented as part of the 21st USENIX Security Symposium (USENIX Security 12). 2012: 569-584.  [20] Kirat D, Vigna G, Kruegel C. Barecloud: bare-metal analysis-based evasive malware detection[C]//23rd USENIX Security Symposium (USENIX Security 14). 2014: 287-301.  [21] Tam K, Khan S J, Fattori A, et al. CopperDroid: Automatic Reconstruction of Android Malware Behaviors[C]//NDSS. 2015.  [22]Poeplau S, Fratantonio Y, Bianchi A, et al. Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications[C]//NDSS. 2014, 14: 23-26.  [23]Backes M, Bugiel S, Schranz O, et al. ARTist: The Android Runtime Instrumentation and Security Toolkit[J]. arXiv preprint arXiv:1607.06619, 2016.  [24]Sun M, Wei T, Lui J. TaintART: A Practical Multi-level Information-Flow Tracking System for Android RunTime[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 331-342. | | | |

|  |
| --- |
| 1. **研究内容和目标（说明课题的具体研究内容，研究目标和效果，以及拟解决的关键科学问题。此部分为重点阐述内容）（不少于2500字）**   本次研究是针对日益严重的安卓应用窃取用户隐私信息的问题，拟基于动态污点分析技术，对xposed框架进行修改，以适应安卓系统新特性，实现对安卓系统中所有app的动态监测，并设计自动化测试模块和防御模块，以实现对大批量app的自动化测试和对app窃取隐私信息的防御功能。主要包括三部分的内容：安卓新特性下的对xposed框架的定制、自动化测试工具和隐私窃取防御模块。  2.1 安卓新特性下的对xposed框架的定制  2.1.1 安卓系统新特性  Google从安卓4.4时引入ART运行时作为一个备选项，在安卓5.0时使用ART彻底替代了Dalvik虚拟机，并且在随后推出的安卓6.0中将默认的编译器后端从Quick模式变为了Optimizing模式，并且支持64位处理器。  Dalvik虚拟机曾经是安卓系统的核心组成部分之一，自Android 2.2开始，Dalvik支持JIT（just-in-time，即时编译技术）。在Dalvik环境下，应用每次运行的时候，字节码都需要通过即时编译器转换为机器码，这会拖慢应用的运行效率，而在ART环境中，应用在第一次安装的时候，字节码就会预先编译成机器码，使其成为真正的本地应用。这个过程叫做预编译（AOT,Ahead-Of-Time），这样的话，应用的启动（首次）和执行都会变得更加快速。但是由于曾经的动态污点分析技术全部需要依赖Dalvik虚拟机的解释器来进行污点传播和检测，因此当AOT替换JIT成为安卓默认的编译方式时，曾经的动态污点分析技术对于ART环境都不再适用。  dalvik与ART区别  图1 Dalvik和ART对比图  如图1所示，app的源代码经过javac或者dx转变为dex字节码即classes.dex文件，Dalvik和ART的输入均为dex文件，但是Dalvik环境中，在app安装时仅进行了dex字节码的优化，生成odex文件，在app运行时将odex文件加载到Dalvik虚拟机中进行解释运行。而ART环境中，在app安装时通过dex2oat编译器将dex字节码转换为编译好的机器代码-oat文件，在app运行时无需进行字节码到机器码的解释过程。OAT文件是一种安卓私有的ELF文件格式，它不仅包含从dex文件翻译而来的本地机器指令，还包含有原来的dex文件内容。  ART结构图  图2 dex2oat的工作流程  图2所示，为ART环境中的dex2oat编译器的工作流程图，dex2oat的输入仍旧是dex文件，它首先检验dex文件的有效性，然后dex2oat将整个的编译过程委托给了后端，它只负责将编译的结果和原始的dex代码写入oat文件。安卓系统中允许的后端有两种模式，一种是QUICK模式一种是Optimizing模式，但是在安卓6.0中，Google将安卓5.0中默认的Quick后端模式替换成了Optimizing模式。后端需要完成的工作分为三个步骤：①转换：将dex代码转换为中间表示形式IR；②优化：对中间形式IR进行代码优化，比如去除冗余指令等；③代码生成：使用代码生成器将IR转化成本地代码。为了实现动态污点分析，需要对ART的dex2oat编译器进行改进，使其允许污点的传播和检测。  2.1.2 xposed框架  Xposed框架是安卓平台上最负盛名的一个框架，它通过替换/system/bin/app\_process程序控制了zygote，使得app\_process在启动过程中会加载XposedBridge.jar这个jar包，从而完成对zygote进程及其创建的Dalvik虚拟机的劫持，与传统的Inhook方式相比，Xposed在开机的时候完成所有的HookFunction的劫持，在原Function执行的前后加上了自定义的代码。它主要分为三个部分：①Xposed：Xposed的C++部分，主要是用来替换/system/bin/app\_process，并为XposedBridge提供方法；②：XposedInstaller：Xposed的安装包，负责配置Xposed工作环境并且提供对基于Xposed框架Modules的管理，在安装XposedInstaller之后，app\_process与XposedBridge.jar放置在了/data/data/de.robv.android.xposed.installer；③XposedMods：使用Xposed开发的一些Modules，其中AppSettings是一个可以进行权限动态管理的应用。  从上述对Xposed框架的介绍中也可以看出，Xposed框架也是依赖于Dalvik虚拟机的，但是Xposed框架的开发者历时一年对该框架进行了调整，使其可以适配安卓5.0和安卓6.0出现的ART运行时新环境，具体的措施如下：①XposedInstaller用定制的app\_process替换了/system/bin/app\_process，对于ART运行时，XposedInstaller也替换了成了定制的ART运行时；②XposedInstaller在定制的app\_process中激活了xposed::initialize()来开启hooking；③定制的ART运行时（Xposed/android\_art）中增加了EnableXposedHook()方法；④EnableXposedHook()通过替换ART运行时汇总的ClassLinker中的ArtMethod pointer来替换方法指针。即适配了ART环境的Xposed框架对ART运行时进行了修改，但是没有对ART编译器即dex2oat进行大规模的改动，而要完成动态污点的传播和监控，就需要对dex2oat进行相应的插桩，因此如何在不影响Xposed框架功能的情况下实现对dex2oat的修改以能够进行动态污点分析是需要研究的重点问题。  2.1.3动态污点分析技术  动态污点分析技术的几个关键要素是污点源（敏感函数中的敏感数据（用户隐私数据）），污点标签（追踪敏感数据而添加的标签），污点标签存储（对数据的污点标签状态的存储），污点槽（污点数据会从某些特定的函数离开系统，这些函数被称为taint sink）和污点逻辑（污点的传播和消除）。  目前安卓系统上的隐私信息即污点源可以分为五类：①设备信息，如IMSI,IMEI,ICCID和SN，这些污点源都在类TelephonyManager中；②传感器数据，如Accelerometer和Rotation，这些污点源在类SensorManager中；③地点位置信息，如GPS，Network Location等，这些污点源在类LocationManager中；④敏感内容，如短信/彩信/通讯录/通话记录等在ContentResolver中，而文件内容在File中，相机在Camera中，麦克风信息在MediaRecorder中；⑤系统信息，如已安装应用和其活动，系统活动日志等。  传统的安卓系统上的动态污点分析技术会对DVM方法参数和方法变量进行变量级污点跟踪机制，在应用程序之间使用消息级别跟踪，对于系统提供的原生库使用方法级别跟踪，最后使用文件级别跟踪以确保永久性信息能够以保守的方式保留其污点标志。由于ART只是涉及app的安装编译和运行方面，所以对于安卓6.0来说，需要重点关注变量级别的污点跟踪机制，其他的例如消息级别跟踪和文件级别跟踪都可以采用以前的技术来实现。  对于污点标记存储而言，传统的污点跟踪系统为每个数据字节或字存储一个标记，在实现中通常的做法是标记包含单独一个比特位，为了降低存储开销，会采取非相邻影子内存或标记图来维护污染字节的标记。TaintDroid采取了仅持续追踪变量的污点状态来显著缩小要跟踪的内存范围的做法，这样就可以将污点标记存储在与变量相邻的内存中。TaintART采取的则是将污点状态存储在处理器寄存器中的做法，这样处理速度更加快且占用的存储空间会更小。Xposed框架下想要采用同样的处理器寄存器来存储污点状态的方法需要对64位处理器的寄存器进行更加深入的研究  2.2自动化测试工具  当前常用的开源安卓自动化测试工具有Robotium、MonkeyRunner、Appium等，它们都各有各的特点，比如，Robotium：适用于为不同的安卓版本和子版本测试进行自动化，但是修改程序源代码，且不适合与系统软件的交互，不能锁定和解锁智能手机或平板电脑；MonkeyRunner：不必处理源代码来做自动化测试，并且可以连接到电脑或模拟真实设备运行测试，但是必须为每个设备编写脚本，每次测试程序的用户界面变化都需要调整测试脚本。因此需要对这些自动化测试工具进行一定的整合和改进，使其符合我们对于大规模测试app的需求。  2.3 防御措施  对于安卓应用的隐私窃取行为，常用的防御措施有发送伪数据，限制应用对敏感信息获取的权限等。拟采取的措施是针对不同级别的隐私窃取对用户做出相应的提醒，并为用户提供是否发送伪数据的选项。  2.4本课题的研究目标和效果如下：   1. 完成一套定制安卓系统，实现对手机上所有APP的动态监控，在app外传信息时分级别对用户进行提醒并记录在文件中，且提供是否发送伪数据的选项   （2）在CCF列表会议或期刊上发表论文1篇；或者申请国家专利1项。 |

|  |
| --- |
| **三、研究方案设计及可行性分析（包括：研究方法，技术路线，理论分析、计算、实验方法和步骤及其可行性等）（不少于800字）**  本项目拟实现基于动态污点分析技术的针对安卓新特性的对窃取隐私信息的安卓应用的检测和防御，该过程主要包括xposed框架的修改、对大规模app进行的自动化测试、隐私信息窃取防御三个部分。   1. 安卓新特性下对xposed框架的修改   安卓6.0现在支持64位处理器，针对动态污点分析技术，以32位处理器为例，用一个32位处理器寄存器（假定R5）进行污点状态存储，用一个bit来表示两个污点状态，这样32位的寄存器中的bit0-bit15可以用来存储16个寄存器（R0到R15）的污点标签状态，bit16-bit31用来存储16个浮点数寄存器（S0到S15）的污点标签状态。利用ART编译器的寄存器分配器来保证污点状态存储寄存器R5不会被分配做其他用途，比如变量存储等。  由于处理器的寄存器数量有限，所以有时寄存器分配器会临时性地把变量存储在主存中即寄存器溢出。而如果寄存器中的变量被移到内存，那么寄存器的污点标签就不再有效，所以拟将污点标签也存进内存，就在被移动的变量后，这样可以保证寄存器溢出后的污点标签仍然能够继续传播。对于ARM64架构，它提供31个常用寄存器，可以使用其中的三个寄存器来存储污点状态，追踪24个数据存储的寄存器。而对于研究内容中提到的5种隐私信息分类，则可以使用3个bit来表示5种不同的状态，这样也可以实现分级对用户进行提醒。  对于方法调用级别的污点传播，利用方法的参数来进行污点传播，而对于IPC级别和文件级别的污点传播，由于这两个级别的污点传播不受ART运行环境的影响，所以采用现存的方法进行处理。  拟在Xposed框架基础上对dex2oat编译器进行插桩，对中间形式IR进行代码插入以实现上述的污点传播逻辑，并且由于xposed框架可以不修改系统代码就hook系统API，所以可以较简单的对source进行污点标记和对sink函数进行检测。   1. 自动化测试和防御模块   安卓现有很多开源的自动化测试软件，选取较适合本课题的自动化测试工具进行改造，使其可以满足课题的需求，实现对大规模安卓应用的检测。而对于发送伪数据进行隐私信息泄露的防御措施，可以利用Xposed框架的模块化特性，借助对系统API的hook来编写可以伪造设备数据的模块。 |

|  |
| --- |
| **四、本研究课题可能的创新之处（不少于500字）**  本研究项目主要的创新点主要有：①基于Xposed框架，在安卓6.0系统上实现了针对隐私信息窃取检测和防御的系统级支持；②针对大规模安卓应用实现了自动化测试。  ①针对隐私信息窃取检测和防御的系统级支持  动态污点分析技术是一个行之有效的动态分析方法，也被大量运用在安卓4.4以前的系统上，且进行了大量的实验，但是Google在安卓5.0和6.0上推出的ART运行时对动态污点分析技术提出了新的挑战。若要利用动态污点分析技术，则需要对ART编译器和运行时进行相应的改进。  拟采用了安卓系统中相当著名的框架Xposed框架，且对框架本身进行修改，而不只是编写一个模块，这样在保持了Xposed框架可扩展性的同时，为框架本身添加了动态污点分析的功能。现有的基于Xposed框架的工作均只是利用了该框架编写模块而不是对框架本身进行改进，本课题基于Xposed框架针对安卓的新特性诸如ART运行环境和64位处理器等进行相应的修改，从而可以利用动态污点分析技术来实现针对隐私信息窃取检测和防御。  ②针对大规模安卓应用实现了自动化测试  目前实现在适应安卓新特性的动态污点分析技术基础上，还增加了自动化测试模块，用户除了可以随时监测本机上在使用的应用外，还可以通过自动化测试工具来对大规模安卓应用进行检测。目前已有的针对安卓6.0新特性的动态污点分析系统均没有自动化测试模块。 |
| **五、研究基础与工作条件（1.与本项目相关的研究工作积累基础 2.包括已具备的实验条件，尚缺少的实验条件和拟解决途径）（不少于500字）**  实验室在国内较早从事安全操作系统相关的研究和原型开发，曾经承担过系统安全、软件漏洞分析方面的项目，发表了高质量的论文。实验室对基于Linux内核的系统的安全性有较为深入的研究，如参与了某安全操作系统的核心安全模块开发，参与编写了安全操作系统规范，对Android系统进行了基于强制访问控制（MAC）的安全增强。实验室对系统安全漏洞也进行了相关研究，如发现了Linux内核源码2处代码缺陷，开发了软件缺陷静态分析工具ABAR，该工具可以分析C/C++程序，并使用此工具发现了FreeBSD系统中10多处代码安全缺陷。另外， 实验室成功的将Linux内核的强制访问控制（MAC）机制引入到虚拟机部署中，实现QEMU/KVMK平台上部署虚拟机间的强隔离，并建立基于安全标签的虚拟机安全迁移通道，构建安全的私有云IaaS平台。实验室利用KVM内核虚拟化技术提供的安全隔离环境，实现操作系统中应用程序的安全隔离，为用户提供透明的应用程序使用接口。针对安卓系统，实验室在Android系统上实现应用个性化和隐私保护的系统级支持，并对第三方提供的分类器应用进行安全审查。  本人之前对安卓系统的发展和漏洞缺陷有较为深入具体的学习，对Android系统的内核和应用架构层有深入的理解，对Android应用程序的静态分析和动态分析技术有较为深入的学习，同时参与了对RTuinOS和FreeRTOS的软件验证工作。同时，本人通过对国际上相关项目的长时间关注，已基本了解并掌握了针对本项目的一些关键技术和算法的现有研究成果。  通过论文组的讨论和验证，在具有以上基础的前提下本论文的研究内容和方案切实可行。 |

**学位论文工作计划**

|  |  |  |
| --- | --- | --- |
| 时间 | 研究内容 | 预期效果 |
| 2016.11—2017.1 | 调研课题相关内容，做好准备工作 | 基础知识扎实 |
| 2017.2 | Xposed框架定制 | 系统概要设计报告 |
| 2017.3—2017.6 | 系统实现 | 系统详细设计报告，实现完整的系统 |
| 2017.7 | 系统测试和实验评价 | 系统测试及评价报告 |
| 2017.8 | 撰写小论文 | 投稿小论文 |
| 2017.9—2017.10 | 根据投稿反馈情况，修改论文；也可能修改系统，补充实验数据 | 发表小论文 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 评  定  小  组  成  员 | 姓 名 | 职 称 | 单位名称 | 职务 |
| 芦效峰 | 副教授 | 北京邮电大学 | 组长 |
| 梁洪亮 | 副教授 | 北京邮电大学 | 成员 |
| 谭咏梅 | 副教授 | 北京邮电大学 | 成员 |
| 肖达 | 讲师 | 北京邮电大学 | 成员 |
|  |  |  |  |
|  |  |  |  |
| 导师意见： | | | | |
| 1、 选题的价值和意义评价  该论文对基于缺陷模式的静态分析漏洞挖掘工具进行研究。随着计算机技术的飞速发展及软件的普及，人们对软件的安全性、可信性也提出了越来越高的要求。如何提高静态分析技术的准确性，减少误报和漏报以确保软件质量及软件的可信性，已成为日益关注的热点。  2、 研究内容的技术可行性评价  实验室缺陷分析工具abazer提供了很好的静态分析技术框架。目前，在静态分析领域，路径切片算法已得到学术界各位学者的拓展并应用到实际，符号执行技术已被应用到很多静态分析工具中，已经相当成熟。故该论文的技术方案是可行的。  3、 论文成果要求和考核指标  成果要求如下：在CCF列表的会议或期刊发表文章一篇。或者为研发的系统申请专利一个。  技术指标如下：拟扩展缺陷模板，提高漏洞检测的自动化能力；拟编写基于缺陷模式的路径切片算法，减少状态空间的爆炸；拟引入符号化分析方法，消除不可达路径以便提高静态分析的准确性。  综上，同意开题。 | | | | |
| 导师（签名）：  日期： 年 月 日 | | | | |
| 开题报告小组意见： | | | | |
| 组长（签名）：  日期： 年 月 日 | | | | |
| 学院意见（签章）： | | | | |
| 负责人：  日期： 年 月 日 | | | | |