■ doi ： 10.3969/j.issn.1671-1122.2014.12.005

基于沙盒的 Android 恶意软件动态分析方案

赵洋，胡龙，熊虎，秦志光

（电子科技大学计算机科学与工程学院，四川成都 611731）

摘　要：智能手机的普及极大地刺激了恶意软件的广泛传播，Android 平台因其巨大的市场占有率和开源特性，已成为攻击者首选的攻击目标。针对传统的基于签名的反病毒软件仅能检测已知恶意软件的缺点，文章提出基于沙盒的 Android 恶意软件动态分析方案，用于有效地分析未知恶意软件的行为。文章通过在虚拟化软件 Oracle VM VirtualBox 中安装 Android x86 虚拟机的方式来实现 Android 沙盒，利用 VirtualBox 提供的命令行工具来控制 Android 沙盒。Android 应用程序通过调用相应系统 API 来完成对应的行为，文中方案通过在应用程序包中插入 API 监视代码的方法监测 Android 应用程序调用的系统 API，并通过脚本程序向 Android 沙盒发送不同的用户事件流来模拟用户对应用程序的真实操作，控制 Android 应用程序在沙盒中自动运行，实验证明文中提出的方法切实可行。

关键词：Android；恶意软件；沙盒；动态分析

中图分类号 ：TP309　　文献标识码：A　　文章编号 ：1671-1122（2014）12-0021-06

中文引用格式：赵洋，胡龙，熊虎，等. 基于沙盒的 Android 恶意软件动态分析方案[J]. 信息网络安全，2014,(12)：21-26.

英文引用格式：ZHAO Y, HU L, XIONG H, et al. Dynamic Analysis Scheme of Android Malware Based on Sandbox[J]. Netinfo Security, 2014, (12):21-26.

Dynamic Analysis Scheme of Android Malware Based on Sandbox

ZHAO Yang, HU Long, XIONG Hu, QIN Zhi-guang

(*School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 611731, *China*)

Abstract: The popularity of smart phones have greatly stimulated the spread of malicious software, because of its huge market share and revenue characteristics, the Android platform has become the preferred target of attackers. Since the traditional signature-based antivirus software can effectively detect known malicious software, the unknown malware is powerless. In this paper, we proposed a novel dynamic analysis scheme of Android malware based on sandbox, which is used to analyze unknown malware effectively. The scheme implements the Android sandbox by installing Android x86 virtual machine in the virtualization software Oracle VM VirtualBox, while using a command-line tool provide by VirtualBox to control the Android sandbox. The Android application performs the corresponding action by calling the appropriate API. We determine the behavioral characteristics by monitoring the API information called by Android application. We make the Android application to run automatically by inserting monitoring codes in the application package and transmit

收稿日期 ：2014-07-14

基金项目：国家自然科学基金 [61003230，61370026]；四川省应用基础研究计划 [2014JY0041]；广东省产学研重点项目 [2012B091000054] 作者简介：赵洋（1973-），男，四川，副教授，博士，主要研究方向：网络安全；胡龙（1988-），男，江苏，硕士研究生，主要研究方向： 网络安全；熊虎（1982-），男，四川，副教授，博士，主要研究方向：网络安全；秦志光（1956-），男，四川，教授，博士，主要研究方向： 网络安全。

通讯作者：胡龙 hulong.hl@gmail.com

different user flow of events to simulate real operations of users on the application. Experiments show that the proposed scheme is feasible.

Key words: Android; malware; sandbox; dynamic analysis

0 引言

据全球著名的信息技术、通信行业和消费科技市场研究机构 Strategy Analytics 的报告显示，2013 年第四季度 Android 的市场份额高达 78.4%，相比 2012 年同期的 70.3% 进步显著[1]。由于 Android 平台巨大的市场占有率和开源特性，以及在巨大的经济利益的驱使下，众多的攻击者开发了大量的针对 Android 平台的恶意软件。早在 2008 年 9 月，安全研究小组 Blitz Force Massada 首次完成了针对 Android 平台的攻击[2]。

Android 平台的恶意软件检测方式主要分为动态(dynamic) 和静态(static) 两种。传统的静态分析方法主要采用基于签名(signature-based) 的检测方案，无法检测未知的恶意软件。而且，基于单一签名的检测方法被加壳、代码混淆等技术所规避。因此，必须保证每一个恶意软件变种的特征签名，这导致特征库不断地膨胀，进而使病毒特征检索复杂度变大，最终带来能耗的增加。

动态检测通常利用沙盒、虚拟机来模拟软件的执行过程，从而获取软件在运行过程中的各种行为、调用的系统 API 以及网络访问等。Gianluca Dini 等人[3] 提出了针对 Android 恶意软件的多级异常检测方法(multi-level anomaly detector for Android malware，MADAM)，该方法同时监视 Android 的内核级和用户级，使用机器学习技术来区分正常行为和恶意行为。在文献[4] 中，Justin Sahs 等人提出了一种用机器学习来检测 Android 恶意软件的方法，通过开源项目 Androguard[5] 从 Android 应用程序包 (APKs) 中提取特征，采用支持 LIBSVM[6] 接口的 Scikit-learn 框架[7] 提供的二分类支持向量机训练这些特征并生成一个分类器，用来分类 Android 软件。基于行为的检测方法则依靠软件的行为(例如，通过动态拦截或者静态分析的方法获取程序的系统调用序列)，结合已知的恶意行为模式，来判断应用程序是否具有恶意行为[8,9]。基于行为检测的方法有三个优点：特征库较小，无需频繁更新；可以预防未知病毒；能够抵抗混淆和加密攻击[10]。Iker Burguera 等人[11] 提出了基于行为的 Android 恶意软件检测系统 Crowdroid，Crowdroid 提供了一个框架来区分具有相同名称和版本、不同行为的应用程序，达到检测具有木马特性的恶意软件的目的。在文献[12] 中，Michael Grace 等人提出了一个主动识别零日 Android 恶意软件的方案，他们把潜在的风险分为三类，高风险、中等风险和低风险；风险评估分为两个阶段，第一阶断的风险评估的目标是直接确定应用程序属于高风险或者是中等风险，第二阶断的风险评估致力于发现应用程序的可疑行为。

上述所有的方案都没有为 Android 恶意软件提供一个真实的运行环境，暴露其真实的行为目的。本文提出的方案正是为了弥补这种不足，为待检测软件提供一个运行环境，通过对 APK 包进行预处理来定制 Android 应用程序在运行过程中输出的 API 调用信息，结果更加真实可靠地反应其在实际使用过程中的行为目的。

沙盒是国际反病毒业界近年来提出的反病毒新概念。它在计算机系统内部构造一个独立的虚拟空间，当发现程序的可疑行为时让程序继续运行，确认是病毒时才终止。沙盒让程序的可疑行为充分表演，同时记下它的每一个动作，在病毒充分暴露了其病毒属性后，沙盒则会执行“回滚”机制，将病毒的痕迹和动作抹去，使系统恢复到初始状态[13]。

本文提出了一个全新的基于沙盒的 Android 恶意软件动态分析方案，设计并实现了一套完整的恶意软件动态分析框架，对未知的恶意软件具有良好的分析效果。本文的主要工作包括：

1) 设计并实现一个 Android 沙盒，提供 Android 应用程序的运行环境，同时 Android 沙盒还具有可恢复性，在每次分析任务结束后，能“回滚”到初始状态。

2) 对 Android 软件进行预处理，反编译 APK 文件，对需要监视的 API 进行“插桩”，在软件执行过程中，凡是被“插桩”的 API，一旦被调用，都会及时输出到日志中。

3) 设计能够控制 Android 软件在沙盒中自动运行的脚

本程序，尽可能多地暴露被检测软件的各种行为（如网络行为），以便获取软件调用的所有API。

1 基于沙盒的Android恶意软件动态分析方案

本文基于沙盒的Android恶意软件动态分析方案采用如图1所示的系统架构，系统分为服务器端和客户端两个部分，用户通过客户端提交任务，服务器接收任务后，对任务进行预处理，然后把任务分发给由Android x86虚拟机组成的沙盒系统，通过自动化控制脚本控制Android应用程序在沙盒中自动运行，根据需要获取应用程序在运行过程中调用的API。此外，服务器主机与Android x86虚拟机之间是通过虚拟网络进行通信的，Android x86虚拟机采用桥接的网络连接方式，这样可以保证Android x86虚拟机与服务器主机之间的双向通信[14]。该方案主要由任务预处理、任务调度和自动化动态分析三个模块组成。

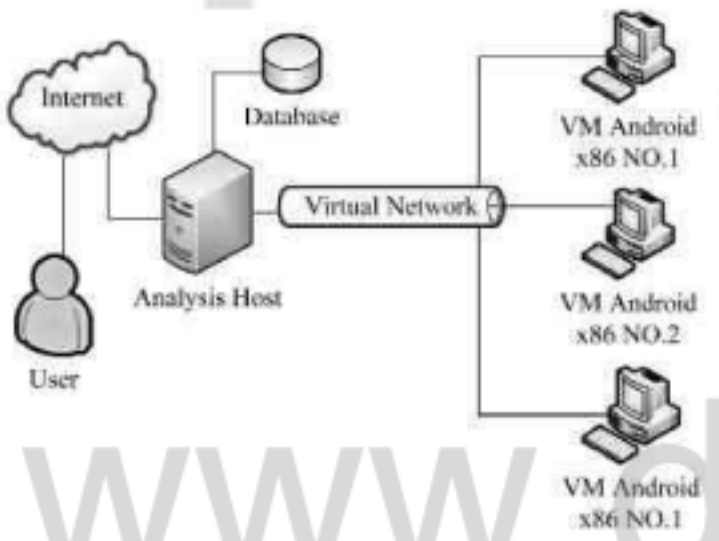

图1 系统架构图

1.1 任务预处理模块

任务预处理模块负责任务的初始化工作。在APK文件安装到沙盒之前，需要对APK文件进行预处理，在APK包中注入监控代码，监控应用程序在运行过程中所调用的API。注入监控代码后，对APK重新打包，把该任务加入任务数据库，等待任务调度模块调度。本模块工作流程如图2所示。

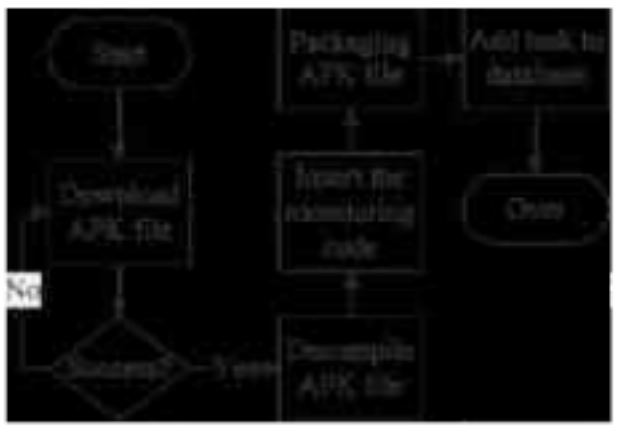

图2 任务预处理模块流程图

1.2 任务调度模块

任务调度模块会一直监控任务数据库，一旦发现新任务，就会开启一个新的线程，该线程负责控制Android x86虚拟机沙盒的运行，直到分析任务结束[15]。由于本文基于沙盒的Android恶意软件动态分析方案支持多任务处理，所以任务调度模块需要具备多线程管理的能力，监控多个沙盒的运行状态。本模块工作流程如图3所示。

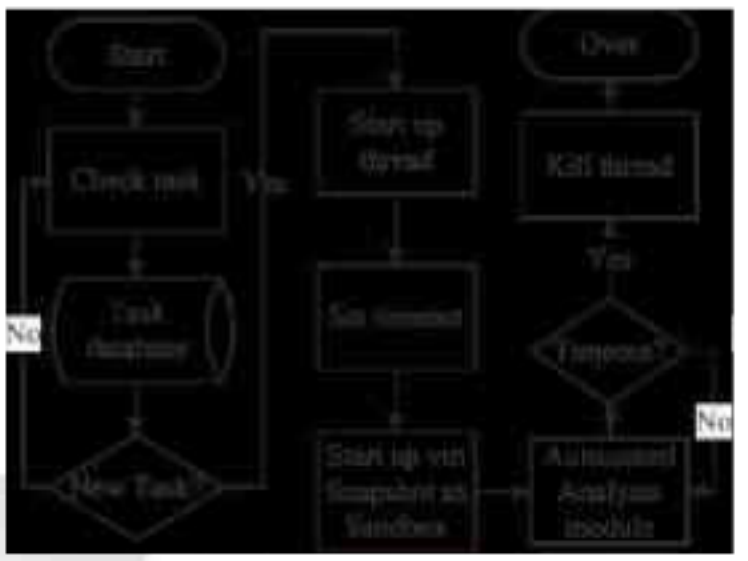

图3 任务调度模块流程图

1.3 自动化分析模块

自动化分析模块是本文的重点，如何控制应用程序在沙盒环境中自动运行，如何触发更多的应用程序行为，都是本模块需要重点解决的问题。

自动化分析模块把经过任务预处理模块处理后的APK包自动安装到沙盒系统中，并启动该应用程序，利用自动化控制脚本程序，模拟用户对软件的操作，控制应用程序自动运行，尽可能多地暴露应用程序的恶意行为，获取应用程序调用的API信息。自动化分析模块模块工作流程如图4所示。

图4 自动化动态分析模块流程图

2 基于沙盒的Android恶意软件动态分析系统实现

2.1 沙盒的实现

在计算机安全中，沙盒是一种根据某些安全策略，隔离运行程序的安全机制，常被用来执行未测试的代码、不

www.docin.com

可信的程序等[16]。为了提供Android 应用程序的运行环境，本文设计的沙盒采用Android x86[17] 虚拟机的形式实现。Android x86 是由Cwhuang 和 Beyounn 二人主持设计的开源项目，主要目的是在 x86 平台上提供完整的 Android 解决方案。

本文通过在虚拟化软件 Oracle VM VirtualBox 中安装 Android x86 为 Android 应用程序提供运行环境，实现 Android 沙盒。为了保证沙盒在被恶意软件破坏后系统仍能正常工作，本文巧妙运用了VirtualBox 的快照（snapshot）[18] 功能，把安装并配置好虚拟网络的Android x86 虚拟机沙盒系统做一个快照，此时的快照包含了虚拟机的完整设置，每一个新的任务在启动虚拟机之前，都要先把沙盒恢复到快照时的状态，这样，即使上一次的任务破坏了沙盒，虚拟机虚拟磁盘的内容也可以迅速地恢复到快照时的状态，实现了 Android 沙盒“回滚”的机制。

本文通过调用VirtualBox 提供的VBoxManage 命令实现沙盒的快照、启动、关闭、恢复等操作。一个新的沙盒系统配置完成以后，都会设置一个唯一的 name，其中 name 的命名格式采用 Android_NO.x。为了实现沙盒“回滚”的能力，需要通过 VBoxManage snapshot <name> 命令对其做一个快照，这样每次启动沙盒之前，通过 VBoxManage discardcurrent -state 命令就可以把沙盒恢复到初始的状态，这样即使上一次的任务对其造成了破坏，也可以使其恢复到正常的工作状态。通过调用VBoxManage startvm <name> 命令来启动沙盒以及通过VBoxManage controlvm <name> poweroff 命令来关闭沙盒[19]。

2.2 任务预处理模块

Android 应用程序在运行过程中会调用各种系统API，应用程序通过调用相应的系统API 来完成对应的动作，通过对应用程序所调用的系统API 进行分析，可以推断出应用程序的行为。为了监测Android 应用程序在运行过程中所调用的API，需要对 Android 安装包进行处理来获取API 的调用信息[20]。

本文中通过APIMonitor[21] 向 APK 包中插入监视代码来监测应用程序在运行过程中调用的 API。APIMonitor 首先反编译所要处理的APK 包，接着遍历 smali 代码，如果找到在配置文件中配置的需要监测的 API，则分析这个 API 的参数，插入 DroidBox 包下相应的类的静态函数，最后重新打包这个 APK 包。这样，当应用程序在运行过程中调用到插入了监视代码的 API 后，系统日志中就会出现标签 DroidBox 标记的日志信息，只需过滤出这些日志信息，就可以获得应用程序调用系统 API 的信息[22]。

Android 系统包括各种类型的 API 函数，恶意软件一般通过调用一些特定的敏感 API 函数来实现恶意行为，这些敏感 API 函数包括文件系统操作、网络、通话功能、短信功能等相关函数，如图 5 所示。

图 5 敏感 API 函数

当处理完 APK 包以后，需要把这次任务加入到任务数据库中，等待任务调度模块的调度。其中任务的参数包括APK 包的存储路径、任务分析时间、任务状态等。任务分析时间用来控制Android 应用程序在沙盒中的执行时间，执行时间的长短决定了此次分析任务所能获取到应用程序调用系统 API 的数量。经过大量的实验分析，得到了系统API 调用数量 y 与时间 t 的函数关系，满足 $y=b*\tanh(t/a)$。任务状态有三种，0 代表新任务，1 代表任务已完成，2 代表任务正在执行。

2.3 任务调度模块

本文基于沙盒的 Android 恶意软件动态分析方案支持多任务处理，分析任务的合理调度可以充分发挥系统的性能，任务调度模块利用独立的线程处理每次检测任务。任务调度模块类图如图 6 所示。

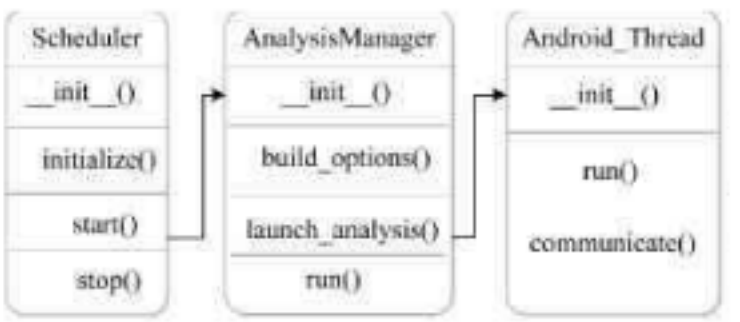

图 6 任务调度模块类图

当系统启动之后，Scheduler 类的 start() 方法会一直循环检测任务数据库，当发现新的任务后，实例化一个 AnalysisManager 对象，并把任务的参数传递给这个

www.docin.com

AnalysisManager 对象。AnalysisManager 对象调用自身的 run() 方法并根据任务的参数初始化相关配置，如 APK 包的路径、分析时间等。接着 AnalysisManager 对象调用 launch_analysis() 方法，首先申请一台空闲的虚拟机，在申请虚拟机之前，要加一个锁，申请到虚拟机之后，释放这个锁。接着实例化 Android_Thread 类，并调用 run() 方法来启动一个 Python Timer 定时器线程，设置任务的分析时间为定时器的超时时间，这样，当达到超时时间后，定时器会自动结束此次任务。

2.4 自动化分析模块

自动化分析模块是本文的重点研究对象。自动化分析模块涉及到应用程序的安装、自动控制和日志获取三个重要的部分。本文利用 Android SDK 工具包提供的 ADB（Android debug bridge）和 Monkey 两个工具来解决这三个问题。

在本系统中，每个沙盒都有单独的 IP 地址，在安装应用程序到沙盒之前，需要通过 adb connect ip:port 命令建立 ADB 与沙盒系统的连接，其中 port 默认为 5554。最后通过 adb install xx.apk 命令把应用程序包安装到沙盒系统中。当任务结束之后，通过 adb disconnect ip:port 命令断开 ADB 与沙盒的连接。

本文利用 Monkey 模拟用户的操作，控制应用程序的自动运行。Monkey 可以运行在实际设备或者模拟器中。本文中，使用 Monkey 向应用程序发送伪随机的用户事件流（如触摸屏输入、按键输入等）来实现对应用程序的操作。

由于不同的应用程序对于不同用户事件流的频率存在较大的差异，而 Monkey 可以向应用程序发送指定的事件，因此本文根据大量的统计信息，获得了不同事件所占的百分比，根据不同事件的出现的频率来定制用户事件流，可以更好地控制应用程序，得到更好的分析结果。表 1 是不同用户事件百分比统计结果。

表 1 用户事件百分比（%）

触摸	12.33
动作	10.97
轨迹	4.91
基本导航	15.23
主要导航	16.52
系统按键	16.47
自定	5.01
其他	4.07

根据表 1 统计的用户事件百分比，本文设计的 Monkey 脚本命令如图 7 所示。

```
def start_monkey(self):
    self.__clean__()
    self.run_cmd("monkey - p <package-name>
                    --throttle 500
                    --pct-touch <percent>
                    --pct-motion <percent>
                    --pct-trackball <percent>
                    --pct-nav <percent>
                    --pct-majornav <percent>
                    --pct-syskeys <percent>
                    --pct-appswitch <percent>"
                )
    return self.__output
```

图 7 Monkey 脚本命令

其中 throttle 控制每次用户事件之间的时间间隔（单位是毫秒），package-name 是应用程序包的名字，percent 是用户事件的百分比。这样，通过修改每种用户事件的百分比，就可以方便地调整控制应用程序的方式。

应用程序在自动运行的过程中，会调用系统的各种 API。本文利用 adb logcat 命令获取应用程序在运行过程中调用 API 的信息，这样就得到了应用程序动态分析的结果。

3 系统测试与分析

本文对 100 种 Android 恶意应用程序进行了测试与分析，其中有代表性的 5 个恶意应用检测结果如表 2 所示。

表 2 Android 恶意软件分析结果

[redacted]	[redacted]	[redacted]	短信	[redacted]	广告
AndroidDogwar.apk		Yes	Yes	Yes	
com.system.dbprocess .apk	Yes	Yes	Yes		
DroidKungFu.apk	Yes	Yes	Yes		Yes
GoldDreamB.apk		Yes		Yes	Yes
AcnetSteal.apk	Yes	Yes	Yes	Yes	Yes

1) AndroidDogwar.apk ：该应用程序读取设备的联系人信息，并且向这些联系人发送短信息，创建 HTTP 连接请求并发送数据到一个远程服务器。

2) com.system.dbprocess.apk ：读写系统安全设置，发送短信息，申请系统 root 权限。

3) DroidKungFu.apk ：应用程序利用漏洞文件获取系统的 root 权限，收集用户的联系人信息，发送到指定的远程服务器。

4) AcnetSteal.apk ：AcnetSteal 接收远程攻击者的指令，收集用户的邮件、电话薄和网络信息。

5) GoldDreamB.apk ：监测电话和短信，窃取设备中的个人信息，此应用中植入了大量的广告并且申请系统 root 权限。

www.doc88.com

恶意软件通常会窃取电话薄、短信、通话记录和邮件等用户信息并发送到远程服务器，部分恶意软件被植入了大量的垃圾广告，攻击者通过远程指令控制设备，窃取用户的隐私信息。通过对恶意软件进行大量分析，本文得到了各种恶意行为在恶意软件中所占的百分比，如表3所示。

表3 Android软件恶意行为百分比（%）

行为	百分比
短信	77.0
网络	81.0
电话簿	67.0
系统设置	17.0
超级权限	26.0
广告	24.0
其他	30.0

4结束语

本文对Android恶意软件检测技术进行了系统的调研，分析了每种技术的特点和存在的不足，在此基础上提出了一种全新的基于沙盒的Android恶意软件动态分析方案。本文提出的方案解决了现有方案存在的不足，为Android应用程序提供了一个真实的运行环境，使应用程序暴露出其真实的行为。

本方案也存在几点不足之处。首先，对Android沙盒的控制依赖于虚拟化软件，虚拟化软件的稳定性决定了Android沙盒的稳定性；其次，Android沙盒系统的版本依赖于Android x86开源项目，新版本的Android系统如果加入了新的系统API，将无法检测出这些新的API调用；最后，本文的Android沙盒需要同时启动虚拟化软件和Android x86虚拟机，资源开销相对较大。

针对上面提出的三点不足之处，下一步的研究重点是改进Android沙盒，降低甚至消除Android沙盒对虚拟化软件和Android x86开源项目的过分依赖，降低Android沙盒的资源开销。（责编 潘海洋）

参考文献：

[1] Strategy Analytics. Android Captured 79% Share of Global Smartphone Shipments in 2013[EB/OL]. http://blogs.strategyanalytics.com/WSS/post/2014/01/29/Android-Captured-79-Share-of-Global-Smartphone-Shipments-in-2013.aspx, 2014-02-06.

[2] CASTILLO C A. Android malware past, present, and future[EB/OL]. http://www.mcafee.com/us/resources/white-papers/wp-android-malware-past-present-future.pdf 2011.

[3] DINI G, MARTINELLI F, SARACINO A, et al. Madam: a multi-level anomaly detector for android malware[M]. Springer Berlin Heidelberg, 2012: 240-253.

[4] SAHS J, KHAN L. Amachine learning approach to Android malware detection[C]//Intelligence and Security Informatics Conference (EISIC), 2012 European. IEEE, 2012: 141-147.

[5] Anthony Desnos. Androguard[EB/OL]. https://code.google.com/p/androguard, 2014-04-06.

[6] CHANG C C, LIN C J. LIBSVM: a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(03): 27.

[7] PEDREGOSA F, VAROQUAUX G, GRAMFORT A, et al. Scikit-learn: Machine learning in Python[J]. The Journal of Machine Learning Research, 2011, 12(10): 2825-2830.

[8] CHRISTODOTESCU M, JHA S. Static analysis of executables to detect malicious patterns[R]. WISCONSIN UNIV-MADISON DEPT OF COMPUTER SCIENCES, 2006.

[9] LO R W, LEBITT K N, OLSSON R A. MCF: A malicious code filter[J]. Computers & Security, 1995, 14(6): 541-566.

[10] ZHOU Y, JIANG X. Dissecting android malware: Characterization and evolution[C]//Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012: 95-109.

[11] BURGUERA I, ZURUTUZA U, NADJM T S. Crowdroid: behavior-based malware detection system for android[C]//Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011: 15-26.

[12] GRACE M, ZHOU Y, ZHANG Q, et al. Riskranker: scalable and accurate zero-day android malware detection[C]//Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012: 281-294.

[13] JANA S, PORTER D E, SHMATIKOV V. TxBox: Building secure, efficient sandboxes with system transactions[C]//IEEE Symposium on Security and Privacy (SP), 2011: 329-344.

[14] 王马龙，胡晓勤，王琳，等 . Android 智能手机用户场景信息防泄漏系统的研究与实现 [J]. 信息网络安全，2013，(2)：35-37.

[15] 林佳华，任伟，贾磊雷 . Android 手机隐私保护系统的设计与实现 [J]. 信息网络安全，2013，(7)：16-19.

[16] Sandbox[EB/OL]. http://en.wikipedia.org/wiki/Sandbox(computer_security), 2014-04-07.

[17] Android-x86 Project - Run Android on Your PC[EB/OL]. http://www.android-x86.org/, 2014-05-17.

[18] Oracle V M. VirtualBox user manual[EB/OL]. http.//down-load.virtualbos, org/virtualbox/UserManual.pdf, 2014-05-14.

[19] 吴乐华，孙贤鲁，孟槟. 基于Android手机软件认证的U盘锁系统 [J]. 信息网络安全，2014，(3)：68-73.

[20] 李宇翔，林柏钢 . 基于 Android 重打包的应用程序安全策略加固系统设计 [J]. 信息网络安全，2014，(1)：43-47.

[21] droidbox[EB/OL]. https://code.google.com/p/droidbox/, 2014-05-17.

[22] 高岳，胡爱群 . 基于权限分析的 Android 隐私数据泄露动态检测方法 [J]. 信息网络安全，2014，(2)：27-31.

www.doein.com