

Android 安全机制分析与解决方案初探

廖明华 郑力明

(暨南大学信息科学技术学院电子工程系,广州 510632)

摘 要 Android 是 Google 公司推出的手机操作系统。由于其开源、可编程软件框架、网络化设备的性质,Android 易受到智能手机病毒的攻击。从 Linux 机制、Android 特有的安全机制、其它保护机制三个层次全面深入地分析了 Android OS 保护手机安全的机制原理。Android 设备在正常状态下是受到严密保护的,但攻击者很有可能找出某个内核模块或核心库的弱点,进而获得最高访问权限,进行攻击。所以,为进一步强化 Android 设备的安全性,使其能够妥善处理高风险性的威胁,研究了基于主机的入侵检测系统(HIDS)和 SELinux(Security-Enhanced Linux),分别用于检测恶意软件和加强系统底层访问控制。

关键词 Android 安全机制 异常检测 KBTA SELinux

中图法分类号 TN929.5; **文献标志码** A

Android 系统的智能手机市场占有率位居高位,Android 的流行将使 Android 手机成为种类繁多的病毒软件繁殖的肥沃土壤。

Android 最具创新的特点是开放性。这就允许任何一个人都可以开发自己的应用程序并自由地发放。当开放性为开发者和用户提供各种各样的便利时,也加重了安全的忧患。由于缺少对应用程序开发与发放的管制,用户非常可能会下载并安装了网络黑客编写的恶意软件。很明显,这将导致手机部分或全部不能正常使用,莫名的短信或多媒体信息服务的收费,用户个人信息泄露和损害通信网络。所以研究 Android OS 的安全机制,增强防护能力意义重大。

1 Android 简介

1.1 Android 平台框架

Android 是在移动设备上执行应用程序的环境。Android 平台框架共由 5 部分组成:Android 软件栈的基础是 Linux 2.6 内核,它主要提供安全、驱动、内

存管理、进程管理、网络协议栈等核心系统服务。Linux 内核上面是 Android 本地库,是一套 C/C++ 库,被上层各种各样的系统组件调用。在 Android 应用程序内通过 Java 本地调用(JNI)实现合并这些库。Android 运行环境包括 Dalvik 虚拟机和核心库。Dalvik 运行 .dex(Dalvik-executable)文件,一种被认为比 Java 类文件更加简洁和节省内存的文件。核心库由 Java 语言编写,提供了大量的 Java 5 SE 包的子类和一些 Android 特有的库。应用程序框架仍然都是由 Java 语言编写,它是开发者进行 Android 开发的基础。该层主要由 View、通知管理器、活动管理等由开发人员直接调用的组件组成。应用程序也是用 Java 语言编写的,且运行在虚拟机上的程序。整个 Android 平台框架如图 1 所示。

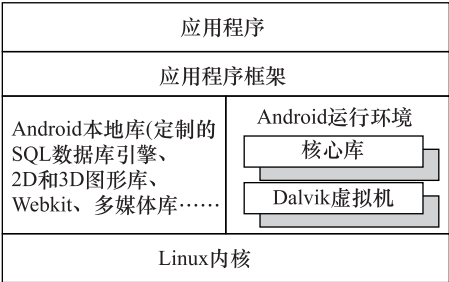


图 1 Android 平台框架图

2011 年 6 月 9 日收到 2010 年粤港关键领域重点突破
项目(2010498E12)资助
第一作者简介:廖明华(1984—),湖南省邵阳市人,硕士研究生,研
究方向:Android 应用开发、多媒体网络通信。

1.2 Android 应用构成

Android 应用程序的常规发布格式是一个经过数字签名的 .apk 文件包,类似于标准的 Java Jar,包含程序所有的代码和非代码文件。其中一个 XML 文件,也就是 Android Manifest 文件,这个布局文件包含应用程序的基本信息,如包名、组件描述、权限声明。

一个 Android 应用程序包含四种类型的组件(子构造块): activity, service, content provider 和 broadcast receiver。Activity 工作在手机屏幕前台与用户进行通信,service 工作在后台没有用户界面。Content provider 为应用程序提供数据存储,broadcast receiver 帮助程序组件之间相互通信。每个组件独立地被实例化和执行的同时也相互作用其它的组件,必要时可以被其它程序启动。

2 Android 安全机制

Google 为 Android 平台配备了多个安全机制。我们将从三个层次进行研究:Linux 机制、Android 特有的安全机制、其它保护机制^[1]。

2.1 Linux 机制

在 Android Linux 内核层包含两个基本的安全机制:可移植操作系统接口 (POSIX) 用户和文件访问控制。这些机制的基本元素是用户(表现为一个整数号码,或者用户 ID),用户拥有对象(如进程和文件),用户再进一步分配到用户组。

2.1.1 POSIX 用户

每个安装在移动设备上的 Android 包(.apk)文件都会分配一个唯一 Linux (POSIX) 用户 ID。所以,两个不同包的代码不能在同一个进程里运行。这就创造了一个沙箱,阻止应用程序对其它程序或系统其它部分施加恶意影响。一个 Android 的应用程序主要运行在一个安全的沙箱内;因此,它不允许访问其它资源,除非明确地授权允许访问。当然,你可以通过设置每个 Android 包的 Android Manifest.xml 文件的 manifest 标签内的 sharedUserId 属性,给它们配备同一个用户 ID。这样,这两个包被

认为是同一个程序,拥有同样的用户 ID 和文件存取权限。要注意的是,为保证安全,只有两个具有相同签名的应用程序(且请求相同的 sharedUserId 属性),才能分配相同的用户 ID。

2.1.2 文件访问控制

Android 文件(包括应用程序文件和系统文件)管理属于 Linux 权限机制。每个文件与用户 ID 和用户组 ID 以及三组 Read/ Write/ eXecute (RWX) 权限密切相关,如图 2 文件属性示意图所示^[2]。

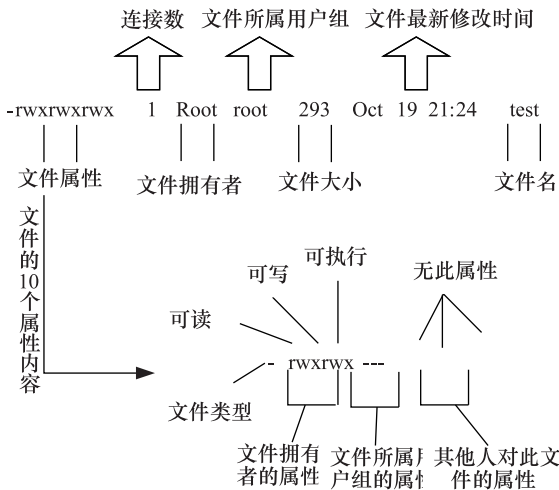


图 2 文件属性示意图

通常,“system”或“root”用户拥有 Android 系统文件,而特定应用程序的用户拥有应用程序的文件。把应用程序和系统文件分配给不同的用户,且设定合适的能保障文件访问安全的权限是有必要的。因为每个应用程序是不允许访问其它应用程序的文件的(除非它们共用同一个用户 ID,或被设定为全局的 readable/writable)。

2.2 Android 特有的安全机制

Google 为 Android 引进了以下特有的安全机制:应用程序权限机制,组件包装,和数字签名。

2.2.1 权限机制

Android 通过在每台设备上实施了基于权限的安全策略来处理安全问题,采用权限来限制安装应用程序的能力。权限是一段唯一的各不相同的字符串。当某个权限与某个操作和资源对象绑定在一起,我们必须获得这个权限才能在对象上执行操

作。Android 框架提供一套默认的权限存储在 android. manifest.permission 类中,同时也允许我们自定义新的权限。我们在写应用程序时声明权限,程序安装时新权限被引入系统。权限授权在应用程序被安装时执行。当在设备上安装应用程序时,程序将请求完成任务必需的权限集合。被请求的权限列单显示在设备屏幕上以待用户审查。只有用户同意授权后,程序才会被安装,该应用程序获得所有被请求的权限。所以,Android 系统实施的主要安全准则是应用程序只有得到权限许可后,才能执行可能会影响到系统其它部分的操作^[3]。

①权限策略

每个权限被定义成一个字符串,用来传达权限以执行某个特殊的操作。所有权限可以分为两个类别:一种是执行程序时被应用程序所请求的权限,一种是应用程序的组件之间通信时被其它组件请求的权限。

开发者通过在 XML manifest 文件中编写权限标签来定义以上两种类别的权限策略。XML 元素 < application > 由一系列代表组件的子元素构成,如 < activity >, < service >, < provider >, 和 < receiver >。

(1) 权限声明:应用程序可以用一个 < permission > 元素来声明权限,用于限制访问特定组件或应用程序。在安装程序时,这个已声明的权限被加入到系统中。

(2) 权限请求:应用程序列出所有需要用来完成任务的权限,分别用 < use-permission > 元素标识。这些权限在程序安装时被请求,列表显示在屏幕上。用户要么同意安装,要么中止安装。同意安装意味着授权所有被请求的权限。

(3) ICC (inter-component communication) 权限保护: < application > 元素和组件元素都有 android.permission 的属性,在这里我们称这个属性分别为应用程序和组件的权限标签。应用程序内的组件可以继承应用程序元素设置的权限标签。当某一组件启动 ICC 时,相关的访问控制器就会查看组件和组件所在应用程序的权限标签集合。如目标组件

的访问权限标签在以上的集合内,允许 ICC 的建立继续进行,否则将会被拒绝,即使这两个组件在同一应用程序内。图 3 描述了该逻辑的进程:组件 A 是否可以访问组件 B 和 C,取决于比较 B 和 C 内的访问权限标签与应用程序 1 内的标签集合的结果。B 和应用程序 1 内都有 i1 标签,所以组件 A 可以访问组件 B,相反应用程序 1 内没有标签 i2,组件 A 不可以访问组件 B^[4]。

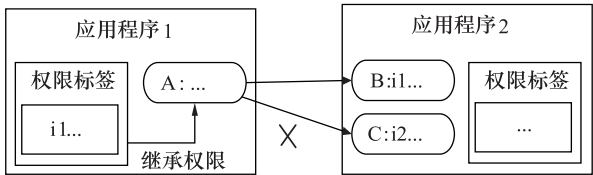


图 3 ICC 访问权限逻辑

②防护等级

防护等级是权限的一个属性,它决定了权限怎样被授权。现今的 Android 安全架构支持四种防护等级: normal、dangerous、signature 和 signatureOrSystem。

(1) Normal: 不是特别危险的权限拥有,在程序安装时,此等权限隐藏在屏幕折叠的菜单内。

(2) Dangerous: 一个可能提供访问私有数据或有害功能的高危险度的权限拥有,如果没有用户的明确证实,应用程序将不会接受这种权限,程序安装时,此等权限明显地罗列在屏幕上。

(3) Signature: 当请求该权限的应用程序与声明该权限的应用程序拥有相同的数字证书签名时,才会授权该权限。

(4) signatureOrSystem: 一种特殊的 Signature 权限,只会授权给安装在 Android 系统镜像的包。

③一个不容忽视的缺陷

Android 权限体制有一些很小但不容忽视的缺点,它表现在以下情况中:

(1) 应用程序可以自由地命名一个新的权限,无须遵循一定的命名规则和限制。期盼用户对不同应用程序(包括未曾安装过的程序)所使用的任意给定的权限名称保持警觉是不够明智的。

(2) 权限一经被授权给应用程序后,在应用程

序的生命期间,它将被不会被移除,即使声明此权限的源程序被删除。

(3) 一个系统内两个不同的权限可以共用相同的名字,以至于他们中的其中一个权限可以在未声明的前提下正常使用^[5]。

2.1.2 组件包装

Android 应用程序(假设它们拥有不同的用户 ID)把组件包装在程序内容内,阻止其他应用程序访问它们。这主要通过定义组件的“exported”属性来实现。如果“exported”属性被设置为“false”,这个组件只能被拥有它的应用程序访问(和其它通过 sharedUserID 属性共用同一个用户 ID 的应用程序)。如果设为“true”,该组件便可以被其它外部程序调用。这些程序的调用受到前面章节提到的权限机制的控制。开发者应该手动地设置“exported”属性值,因为属性的默认值可能与期望的值不一致。

2.1.3 数字签名

所有的应用程序都必须有数字证书,Android 系统不会安装一个没有数字证书的应用程序。Android 程序包使用的数字证书可以是自签名的,不需要一个权威的数字证书机构签名认证。如果要正式发布一个 Android 应用程序,必须使用一个合适的私钥生成的数字证书来给程序签名。数字证书都是有有效期的,Android 只是在应用程序安装的时候才会检查证书的有效期。如果程序已经安装在系统中,即使证书过期也不会影响程序的正常功能。sharedUserId 机制和权限机制都用到签名方法来执行 signature 和 signatureOrSystem 权限。

2.3 其它保护

周围的技术环境(如硬件设备、编程语言、手机载体的基础设施)也提供了一些加强 Android 设备安全性的机制。如内存管理单元(MMU)保障每个用户进程都拥有自己的独立的地址空间,减少了特权升级(Privi-lege Escalation)的可能性;编程语言的类型安全特性能使编写的程序不易受到专制的代码执行(arbitrary code execution)的影响;电话系统的 AAA 系统(authentication, authorization 和 accounting)用于识别用户,监控操作,和向客户收取费

用等。

3 安全解决方案

文献[1]提出许多减轻恶意软件损害移动设备的方法。根据这些方法在 Google Android 上被实施和评估的结果,我们优先选取 SELinux 访问控制机制、入侵检测和 Context-aware 访问控制,而次要选取数据加密、垃圾邮件过滤、可选择性的 Android 权限机制、远程管理,虚拟专用网络(VPN)和身份验证等方案。以下将从基于主机的入侵检测框架(HIDS),SELinux 在 Android 上的实施两个方面探讨 Android 的安全解决方案。

3.1 HIDS (Host-based intrusion-detection system)

在文献[6—9]创新地提出一个在移动设备上检测恶意软件的基于主机的入侵检测系统(Host-based intrusion-detection system),且对其进行评估。该框架(如图4)依靠一个轻型的代理(依据 CPU、内存和电池消耗量)在设备上持续对各种各样的特征取样,采用机器学习(machine learning)和时序推理(temporal reasoning)的方法分析采集的数据,进而推断设备的状态。发送信息、打电话和应用程序等的特征属于应用程序框架类别,通过框架提供的 APIs 提取;而键盘、触感屏幕、时序安排和内存等属于 Linux 内核类别。下面介绍处理器分别为异常检测和 KBTA 的入侵检测系统。

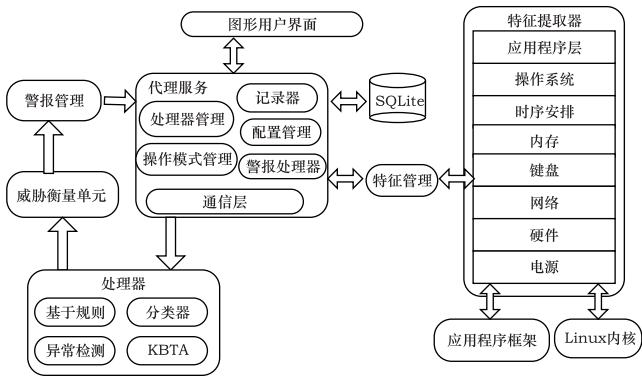


图4 基于主机的入侵检测系统框架

3.1.1 异常检测(Anomaly Detection)

文献[6]中的入侵检测系统框架持续采样各种

各样的系统参数指标,利用机器学习的方法推断设备所处的状态,如所采集的数据是否正常(良性的)或不正常(恶意的)。该方法的主要设想是,通过收集系统参数指标(如:CPU 使用率、通过 Wi-Fi 传送的发送包的数量、运行的进程的数量、电源消耗量等),与已知的恶意软件引发的系统参数指标作对比,检测相同点,进而发现先前未曾遇到的新的恶意软件。因为缺少现成可用的 Android 恶意软件,我们首先开发四个恶意软件,然后评估基于已知恶意软件样品检测发现新恶意软件的能力。我们评估了数个组合实验,组合元素包括不同的分类算法和异常检测算法;不同的特征选择方法;不同的最高特征被选数量。研究的目的在于认识检测算法,特征选择方法,最高特征被选数量怎样区分不包括在训练组内的其它良性软件和恶意软件。当训练和测试在不同的设备上执行,应找到产生最大检测准确度的特定特征。实证结果表明通常这个被提议的框架在检测恶意软件时是有效的,特别在 Android 上(准确率 87.4%,假阳性率 0.126)。

3.1.2 KBTA(Knowledge-based Temporal Abstraction)

在文献[7,8]中,我们检验采用轻型版本的基于知识的时间抽象方法(KBTA)检测恶意软件的适用性,KBTA 可以在资源有限的设备上激活使用。使用 KBTA,结合时间抽象知识基础,持续测量数据(如:运行进程的数量)和事件(如:软件安装)。时间抽象知识基础也就是安全本体,用来从面向时间的原始的安全数据抽象得出高层次的有意义的概念和模式,也可称为时间抽象(如图 5)。

这个新方法适用于在 Google Android 设备上检测恶意软件。评估结果证明了该方法在移动设备上检测恶意应用程序的有效性(大多数情况下检出率为 94%)和在移动设备上运行该系统的可行性(CPU 消耗的平均值为 3%)。

3.2 SELinux(Security-Enhanced Linux)

为了进一步强化 Android 系统和在属于特权用户的关键的 Linux 进程内实施底层的访问控制,文献[10]提议在 Android 内实施 SELinux。采用 SELinux,我们可以更好地保护系统,避免攻击者利

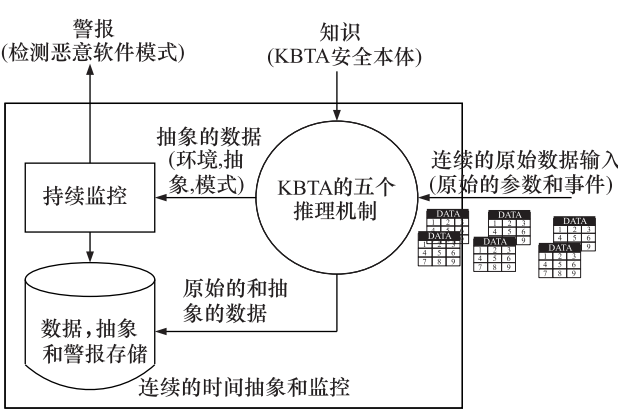


图 5 KBTA 进程

用高特权进程的弱点。我们在 Android 上运行 SELinux,且在 HTC G1 设备上对其性能进行评估。实验表明运行 SELinux 于 Android 设备是可行的,如果可以制定出合适的策略,提升安全等级是可以实现的。这种提升是令人满意的,因为它没有显著的给普通用户带来任何困扰。事实上,用户不需要察觉 SELinux 是否被应用实施。

然而 SELinux 以其笨重的策略维护而闻名,于是我们创立了一个“定向的”策略,它集中于限制已知的关键的进程(例如:系统进程和高特权的守护进程),这就得到了小而轻巧的策略文件。定向策略遗留下来的其它未知的应用程序,将服从其它存在的访问控制机制,如 Android 应用程序权限机制和 POSIX 用户机制。

4 结束语

我们对 Android 框架的安全评估表明恶意软件侵入设备是很有可能发生的,不仅从 Linux 内核层,而且利用 Android 的安全机制从应用程序框架层侵入,所以研发保护 Android 框架的方法是必不可少的。首先,引入一个能够防止或控制源于 Linux 内核层的潜在损害的机制,同样,应该增加更好的保护措施以加固 Android 的权限机制或检测授予权限的误用。Android 源代码的公开可用性使得个人能检查和验证代码。即使 Android 平台不可能完全安全,但开源的途径促使安全性不断地改进。随着时

间的推移,Android 平台的漏洞会越来越少,系统越来越不易受到攻击。

参 考 文 献

1 Shabtai A, Fledel Y, Kanonov U, *et al.* Google Android: a comprehensive security assessment. *IEEE Security & Privacy*, 2010; 35—38

2 鸟哥. 鸟哥的 Linux 私房菜基础学习篇(第二版). 北京:人民邮电出版社, 2007; 96—98

3 Shin W, Kiyomoto S, Fukushima K, *et al.* A formal model to analyze the permission authorization and enforcement in the android framework. *International Symposium on Secure Computing (SecureCom-10)* 2010; 944—945

4 Enck W, Ongtang M, McDaniel P. Understanding android security. *IEEE Security & Privacy*, 2009; 7(1): 53—54

5 Shin W, Kwak S, Kiyomoto S, *et al.* A small but non-negligible flaw in the Android permission scheme. *IEEE International Symposium on*

Policies for Distributed Systems and Net-works, 2010; 109—110

6 Shabtai A, Wiess Y, Kanonov U, *et al.* Andromaly: a behavioral malware detection framework for android devices. *Intelligent Information Systems*, 2011; 7—22

7 Shabtai A, Kanonov U, Elovici Y. Detection, alert and response to malicious behavior in mobile devices: knowledge-based approach. *RAID*, 2009

8 Shabtai A, Kanonov U, Elovici Y. Intrusion Detection on mobile devices using the knowledge based temporal-abstraction method. *Systems and Software*, 2010; 83(8): 1527—1536

9 Shabtai A, Fledel Y, Elovici Y, *et al.* Using the KBTA method for inferring computer and network security alerts from timestamped, raw system metrics. *Computer Virology*, 2009; 8(3): 267—298

10 Shabtai A, Fledel Y, Elovici Y. Securing Android-powered mobile devices using SELinux. *IEEE Security & Privacy*, 2010; 36—44

The Security Mechanism Analysis and
Probe into the Solution of Android OS

LIAO Ming-hua, ZHENG Li-ming

(Department of Electronic Engineering, College of Information Science & Technology, Jinan University, Guangzhou 510632, P. R. China)

[Abstract] As an operating system for mobile device, Google's Android-an open, programmable software framework-is vulnerable to typical smart-phone attacks. To date, there is no effective method available to prevent mobile threats, so mobile security still has a long way to go. Linux mechanism are introduced, Android-specific security mechanism and some other defense mechanisms, which may act as the defense mechanism of mobile security. An Android device in its normal state is well-guarded, however, it is possible for an attacker to identify vulnerability in one of the kernel modules or core libraries, acquire root access and carry out attack. So, to further harden Android devices and enable them to cope with high-risk threats, several security countermeasures are proposed. An overview of some of the most relevant approaches anchored in the area of machine learning is provided, anomaly detection, KBTA, as well as access control using SELinux.

[Key words] Android OS security mechanism anomaly detection KBTA SELinux