文章编号:2095-6134(2015)05-0667-09

一种基于权限控制机制的 Android 系统隐蔽信道限制方法*

吴敬征[1,3†], 武延军[1,3], 罗天悦[1], 武志飞[1], 杨牧天[1], 王永吉[2,3]

(1 中国科学院软件研究所总体部, 北京 100190; 2 中国科学院软件研究所基础软件国家工程中心, 北京 100190;
3 中国科学院软件研究所计算机科学国家重点实验室, 北京 100190)

(2014 年 10 月 11 日收稿; 2015 年 3 月 27 日收修改稿)

Wu J Z, Wu Y J, Luo T Y, et al. A new mitigation approach for covert channel of Android operating system based on permission mechanism[J]. Journal of University of Chinese Academy of Sciences, 2015,32(5):667-675.

摘 要 移动智能终端凭借全新的体系结构、安全机制、丰富的传感设备及应用,在国内拥有近 5 亿台的市场. 然而这些新特性却导致了比经典的攻击行为更复杂的新安全问题——移动智能终端隐蔽信道,泄漏用户隐私. 针对 Android 移动智能终端这种新的复杂环境,目前仍缺乏有效的消除限制方法. 本文将 Android 系统隐蔽信道分成基于共享资源的智能终端隐蔽信道和基于传感器设备的隐蔽信道两种基本模型,并深入研究传感器隐蔽信道的形成机理. 通过对 Android 系统权限控制安全机制的分析,扩展权限控制机制的保护范围,设计和实现了基于权限控制机制的 Android 系统传感器隐蔽信道限制方法. 实验证明该方法在实际的隐蔽信道限制中能够达到限制效果.

关键词 Android; 隐蔽信道; 权限控制机制; Android 传感器; 隐蔽信道限制

中图分类号:TP393 **文献标志码**:A **doi**:10.7523/j.issn.2095-6134.2015.05.013

A new mitigation approach for covert channel of Android operating system based on permission mechanism

WU Jingzheng[1,3], WU Yanjun[1,3], LUO Tianyue[1], WU Zhifei[1], YANG Mutian[1], WANG Yongji[2,3]

(1 *Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*;
2 *National Engineering Research Center for Fundamental Software, Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*;
3 *State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract About 500 million of smart mobile devices have been sold in China because of the new architecture, secure mechanism, rich sensors, and applications. However, the new features cause a new secure problem named smart mobile device covert channel, which is more dangerous than the traditional attacks and leaks users' privacy. At the present stage there are no elimination and mitigation methods. In this work, the Android covert channels are classified into two models including the shared resource-based Android covert channel and the sensor-based covert channel.

* 国家自然科学基金(61303057, 61170072)和核高基国家科技重大专项(2012ZX01039-004)资助

† 通信作者,E-mail:jingzheng08@iscas.ac.cn

The latter one has been carefully studied in this work. The new secure features of permission mechanisms is analyzed and extended to protect the sensors. A new mitigation approach for sensor-based channel is designed and implemented, and the experiments show that the covert channel can be mitigated perfectly.

Key words Android; covert channel; permission mechanism; Android sensors; covert channel mitigation

随着云服务模式和网络接入技术的成熟与发展,移动智能终端凭借其便携的移动特性及丰富的实用功能成为 IT 产业发展新的经济增长点. 工业和信息化部统计数据表明,中国移动电话用户超过 11 亿,其中移动智能终端用户近 3 亿;同时 IDC 数据表明 2013 年智能终端市场出货量超过 3.9 亿台,智能手机用户数将超过 5 亿. 随着商业模式的成熟和用户习惯的养成,智能终端将会承载越来越多的工作、学习、娱乐、社交等日常功能. 然而,在带来便利的同时,移动智能终端尤其是智能手机涉及大量的用户隐私数据,包括用户通讯录信息、短信信息、网上银行信息、地理位置信息、用户日程、用户行为习惯等,这些隐私信息一旦被泄露,将会被恶意用户攻击利用导致直接的经济损失.

典型的攻击行为利用智能终端的安全漏洞和系统实现缺陷,设计漏洞利用程序,从而实施用户隐私窃取、垃圾广告推送、恶意扣费、拒绝服务等攻击行为. 例如,Android 系统中由于 libsysutils 栈缓冲区溢出导致的权限提升漏洞(CVE - 2011 - 3874);Android 系统中 Chrome 浏览器 Download 函数导致的用户隐私信息泄漏漏洞(CVE - 2012 - 4906)等. Zhou 和 Jiang[1] 对 2012 年 2 月采集的 62 519 个 Android 应用程序进行分析,发现 1 279 个信息泄漏和 694 个内容污染风险程序;Grace 等[2] 对 2011 年 10 月期间采集的 118 318 个 Android 应用程序进行分析,发现 3 281 个风险程序,其中 322 个为 0 - day 恶意程序. 有研究者系统总结了 2004—2011 年间的移动智能终端安全威胁,并全面综述了相应的安全解决方案[3-4].

移动智能终端采用全新的体系结构、丰富的传感设备和改进的安全机制,而这些新的特性却导致了比经典的攻击行为更复杂更难检测处理的新的安全问题——移动智能终端隐蔽信道[5-6]. 隐蔽信道是指恶意进程通过合谋信息系统共享资源而实现的一种信息泄漏方式,最早在 1973 年由 Lampson[7] 在程序限制问题的研究中提出,后续的研究扩展到了单机操作系统、数据库系统、网络系统和云计算平台[8]. Soundcomber 是 Android 智能系统中的一款隐蔽信道恶意程序,能够在通话过程中抽取用户隐私信息(如信用卡号、PIN 码等)泄漏给远程服务器[6]. Soundcomber 只需申请录音权限和互联网权限即可在后台秘密地完成信息泄漏,相比普通的恶意软件更加难以检测和处理. TouchLogger 是一款利用智能终端加速计实现信息泄漏的隐蔽信道[9]. TouchLogger 根据移动设备的偏移推测手指触摸按键的位置,从而泄漏用户输入的数字信息.

隐蔽信道分析工作包括信道识别、度量和处置[1]. 信道识别是对系统的静态分析,强调对设计和代码进行分析发现潜在的隐蔽信道,并通过构建信道场景判断该潜在信道是否能够被实际利用[10-11]. 信道度量是对信道传输能力和威胁程度的评价,度量指标包括容量指标[12],隐蔽信道因素,相对容量以及引入传输价值概念的短消息指标[13]等. 信道处置措施包括信道消除、限制和审计[2-3]. 隐蔽信道消除措施包括修改系统、排除产生隐蔽信道的源头、破坏信道的存在条件,或者将危害限制到系统能够容忍的范围内. 但是并非所有的潜在隐蔽信道都能被入侵者实际利用,如果对所有的潜在隐蔽信道度量和处置会产生不必要的性能消耗,降低系统效率. 隐蔽信道检测则强调对潜在隐蔽信道的相关操作进行监测和记录,通过分析记录,检测出入侵者对信道的实际使用操作,为信道度量和处置提供依据[1, 14-15].

由新型传感设备的引入导致的隐蔽信道信息泄漏是隐蔽信道问题在继单机系统、网络、云计算之后在移动智能终端领域的新发展[8]. 到目前为止,国内外对该领域的研究尚处于起步阶段,学术界还没有形成成熟的系统化的研究方案,尚缺乏针对移动智能终端隐蔽信道尤其是 Android 终端隐蔽信道的限制方法.

本文针对 Android 系统这种新型的复杂场景,研究基于传感器的 Android 系统隐蔽信道机

理,并通过对 Android 系统权限控制机制的分析和扩展,设计和实现了基于权限控制机制的 Android 系统隐蔽信道限制方法. 并通过实验证明,该方法在实际的隐蔽信道限制中,能够达到切实可用的限制效果.

1　相关研究

移动智能终端井喷式的发展为信息产业注入了新的活力,其安全性能也受到产业界、学术界和政府各界越来越多的关注. 2012 年,La Polla 等[3]从安全威胁的形式、安全攻击的手段和目的、安全防范方案等角度,全面综述了移动智能终端面临的安全问题,并指出与传统计算机系统相比的 5 个关键区别:随时携带的移动性,用户定制的个性化特性,实时在线性,技术继承性以及资源受限性. 这些差异化特性导致移动智能终端更容易受到隐私泄露、信息嗅探、拒绝服务和恶意扣费等安全问题的影响. 移动智能终端隐蔽信道是一种高级的信息泄漏方式,为全球超 10 亿的智能终端带来巨大的安全威胁[10-11]. 因此,必须对移动智能终端环境下的安全问题,尤其是隐蔽信道问题的机理进行深入研究,才能够为智能终端发展提供更安全的环境.

为保证系统安全和用户隐私,移动智能终端采取了新的安全策略和访问控制模型. 典型的智能终端平台 Android 系统采用“Sandbox”和“Permission”机制为应用提供安全的隔离环境,从而保证应用程序在默认情况下不能执行对其他应用程序、操作系统或者用户有害的操作[12]. 然而,隐蔽信道却能够破坏 Android 系统的隔离性,在不违反安全策略模型的情况下,隐蔽地泄漏用户的机密信息.

2011 年 Schlegel 等[6]提出 Soundcomber 的智能终端隐蔽信道. 在 Android 系统中,Soundcomber 只声明了少量的合法权限,通过控制录音设备对用户通话过程中的上下文及语调信息进行语音识别和特征抽取,识别出对话中涉及的机密信息,如信用卡号、PIN 码、社会保险号以及关键联系人信息等,最终发送到远端服务器,实现隐蔽的用户隐私信息泄漏. 由于 Soundcomber 只访问录音设备,没有其他非法操作,通常的恶意检测方法难以发现和处理. 因此,研究人员提出一种激进的检测方法,在执行敏感通话时,限制一切程序访问音频数据. 这种方法一定程度上限制了 Soundcomber 的威胁,但也会带来相应的副作用.

与 Soundcomber 类似,2011 年 Cai 和 Chen[9]提出另一种利用移动智能终端传感器实现的隐蔽信道 TouchLogger,该信道是在智能终端触摸屏环境下对基于键盘输入的时间隐蔽信道 JitterBugs[13]的扩展. TouchLogger 利用移动智能终端中的动作传感器加速计或者陀螺仪记录用户触摸屏幕时导致的位置偏移和振动反馈信息,结合用户的使用习惯对该信息进行数据挖掘,从而推测用户的输入. 2012 年,Owusu 等[14]对该工作进一步扩展,提出使用智能终端中的加速计传感器进行隐私信息泄漏的隐蔽信道 ACCessory. 与 TouchLogger 不同的是,ACCessory 对屏幕区域进一步细分,从而能够识别更多的用户输入. 2012 年,Miluzzo 等[15]同样提出利用加速计和陀螺仪实现的基于屏幕输入的隐蔽信道 Tapprints. 一系列研究表明,移动智能终端的传感器设备成为隐蔽信道的潜在媒介,其根源在于终端系统对传感设备的访问权限设置不够合理,缺乏针对传感器的控制机制.

对于隐蔽信道问题,国内的研究更侧重于操作系统和网络系统领域,针对移动智能终端隐蔽信道的研究尚无系统化的成果. 王昌达等[16]从审计度量角度对隐蔽信道进行了形式化的分析和描述. 卿斯汉和朱继锋[17]针对安胜 OS v4.0 高安全等级操作系统设计了一种代码层次的标识方法,称为回溯搜索法,成功地发现了 18 条真实的隐蔽信道. 有研究者针对系统源代码的基于有向信息流图的隐蔽信道标识算法分析 Linux kernel 2.6.18 中的 130 万行代码,发现了 43 条隐蔽信道,其中大部分为首次发现;同时将标识的隐蔽信道按特征划分成 4 个典型类型,并分别进行场景分析和威胁限制,证明 43 条检测结果均为真实的隐蔽信道[18-19].

网络隐蔽信道研究方面,2009 年姚立红等[20]研究网络时间隐蔽信道数据包间隔时间与解码错误率之间的关系,将解码过程建模为状态转换模型,从信息论的角度推导出信道容量公式,并将该工作扩展到 Xen 虚拟化平台. 另一种信息网络隐蔽信道引入 Huffman 多元编码机制,从容量和准确性两个方面衡量其威胁,并设计了一种基于并发冲突间隔时间的隐蔽信道检测和限制

方法[21-22].

在云平台隐蔽信道研究中,2011 年 Wu 等[23]对 Xen 虚拟化平台使用基于有向信息流图的方法,首次发现一种基于共享内存的云平台隐蔽信道,并跟踪 Ristenpart 等[24]的研究工作综述了云环境下的隐蔽信道[25]. 该信道利用 Xen 虚拟化平台的域间共享内存机制,实现跨虚拟机的机密信息泄漏. 为限制隐蔽信道的威胁,研究人员从数据机密性和完整性角度提出多种基于安全机制的云平台信息泄漏限制方法,如 sHype[26]、Lares[27]、HyperSentry[28]、HyperSafe[29]、SecVisor[30]等,一定程度上保护了用户的数据安全. 2012 年 Wu 等[31]提出的 Xenpump 限制方法能够限制基于共享内存的云平台隐蔽信道威胁,且证明经过简单扩展,该方法能限制其他类型的隐蔽信道.

2 Android 隐蔽信道模型抽象

隐蔽信道源于程序限制问题,在操作系统、数据库、网络系统和云计算平台中受到了广泛关注. 根据隐蔽信道在不同环境中表现出的特征,研究人员从各自研究的领域和侧重的角度给出了不同的定义,其中 Lampson 和 Tsai 的定义最具有代表性.

Lampson[7]认为如果一条信道既不是设计用于通信的,也不是有意用于传递信息的,就称为隐蔽信道. 虽然隐蔽信道问题最早由 Lampson 提出,但是其定义相对模糊,没有说明隐蔽信道的产生机制和必要因素. Tsai 等[32]对该定义进一步深入扩展:给定一个强制安全策略模型 M 及其在一个操作系统中解释 $I(M)$,$I(M)$ 中的 2 个主体 $I(S_h)$ 和 $I(S_l)$ 之间的通信是隐蔽的,当且仅当模型 M 中的对应主体 S_h 和 S_l 之间的任何通信都是非法的. 比较而言,Tsai 的定义更加全面,指出了隐蔽信道与系统的强制访问控制策略之间的强关联关系,并且指出隐蔽信道需要具备包括两个通信主体的必要条件.

参照 Tsai 的定义,研究人员进一步分析隐蔽信道的传输机制,给出隐蔽信道在操作系统中的定义[32]:隐蔽信道可以形式化表述为 4 元组$\langle V, PA_h, PV_i, P\rangle$,其中 V 表示操作系统中的共享资源;PA_h是修改共享资源 V 的 TCB 原语且具有较高的安全级;PV_i是感知、观察共享资源 V 的 TCB 原语且具有较低的安全级;P 表示强制访问控制模型. 如果从 PA_h到 PV_i的通信是系统安全策略 P 所不允许的,则 PA_h到 PV_i的通信信道称为隐蔽信道.

然而,在 Android 系统中隐蔽信道的形态和特征发生了新的变化. 如图 1 所示,在 Android 智能终端中用户隐私 APP 只声明了用户隐私数据(通讯录、短消息、日程信息)访问权限而没有网络访问权限,在 Android 的“Permission”安全机制保障下,即使该 APP 获得了用户隐私数据也无法向外部泄漏;网络资源访问 APP 只声明了网络访问权限,无法直接读取用户隐私信息;而隐私 APP 通过对共享资源的影响完成隐私信息向网络 APP 的泄漏. 其中的共享资源包括 2 个 APP 可同时读写的 SD 卡文件、CPU 的响应时间、音量大小设置、共享系统日志、共享配置文件等资源. 在基于共享资源的智能终端隐蔽信道中,恶意用户利用 2 个合法的应用程序,合谋实现了终端用户隐私信息的泄漏.

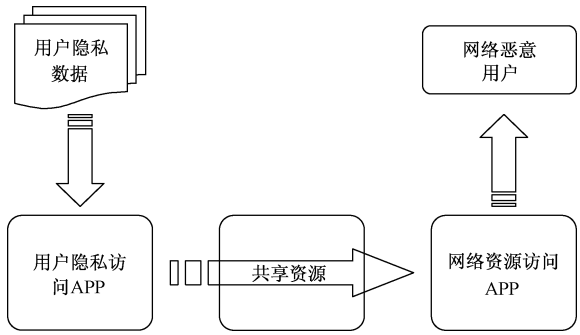

图 1 基于共享资源的 Android 系统隐蔽信道

Fig. 1 Covert channel of Android based on shared resources

与基于共享资源的智能终端隐蔽信道不同,基于传感器设备的隐蔽信道只需利用一个应用程序即可实现用户输入数据的泄漏. 如图 2 所示,该应用程序实时监控终端传感器的采集信息,根据用户的行为进行数据挖掘,从而推测用户输入,最终传输给网络中的恶意用户. Soundcomber、TouchLogger、ACCessory、Tapprints 都属于基于传感器设备的隐蔽信道,由于 Android 系统允许无声明的访问传感器资源(触摸屏、加速计、陀螺仪等),这类隐蔽信道只需申请网络服务权限即可实现隐私泄露,更加难以限制和处理.

3 Android 系统隐蔽信道限制方法

Android 移动智能终端采用全新的安全机制,

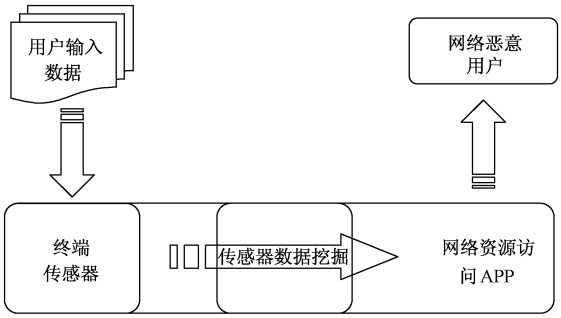

图 2　基于传感器设备的隐蔽信道

Fig. 2　Covert channel of Android based on sensors

即“Sandbox”和“Permission”机制. 其中,权限控制机制是 Android 系统框架中核心的安全机制,它为用户隐私数据保护、设备及软件使用控制、网络使用控制及系统监控、应用程序运行环境隔离等提供保障. 新型的 Android 系统隐蔽信道利用安全机制对特定传感器设备访问控制不完备的漏洞缺陷,导致传感器机密信息泄漏.

与现有的操作系统、网络和云平台隐蔽信道限制技术相比,智能终端隐蔽信道限制的难点在于分析新型安全机制对资源控制的完备性、如何进一步完善 Android 框架层的权限控制机制,从而设计合理的 Android 隐蔽信道限制方法.

3.1 Android 权限控制机制原理

Android 系统的权限控制机制仅分配给应用程序有限的资源,应用程序只能通过受限的系统定义接口访问系统资源. Android 系统通过权限控制机制保护系统资源、限制应用程序能力、防止系统资源被滥用.

定义 1　(Android 权限机制,Permission) Android 权限控制机制可用如下 4 元组及描述公式进行表述:

$< Perm, APPs, Sensors, \rightarrow >, \forall a \in APPs, \exists s \in Sensors,$
If $Perm(a,s) \neq \Theta$, then $(a \rightarrow s) = True$
Else $(a \rightarrow s) \equiv False$

其中,*Perm* 指 Android 系统保护被访问资源的权限. Android 系统定义了大约 130 个默认的权限,应用程序也可以自定义新权限来保护自己的服务. 受保护的传感器 *Sensors* 包括摄像功能、GPS 地理位置、蓝牙功能、电话功能、短信功能、网络功能等.

对受保护资源的访问,应用程序开发者需要在程序配置文件中显式声明所有需要的权限. 如果 Android 应用程序在配置文件中申请了相应的权限,在程序安装时会提示用户该程序的权限申请列表,即 $Perm(a,s) \neq \Theta$. 例如,如果应用程序需要访问 GPS 精确地理位置,则需在配置文件 AndroidManifest. xml 中添加如下声明:

```
< uses-permission android : name = " android. permission. ACCESS_
FINE_LOCATION" / >
```

否则,Android 应用程序程序运行时会产生一个类型为 SecurityException 的异常,并提示缺少权限. 基于权限声明列表,用户选择是否信任该程序并继续安装,安装后应用程序可以对该资源进行相应的访问,即 $(a \rightarrow s) = True$;否则,相应的资源不允许访问,即 $(a \rightarrow s) \equiv False$.

Android 系统的权限控制机制如图 3 所示,申请了 *CAMERA* 权限的应用程序可以访问 Android 手机的摄像头资源和无限制的传感器资源,但是无法访问其他任何未声明的受保护资源;对于未声明任何权限的应用程序只能访问无限制的传感器资源,而不能访问其他任何受保护资源.

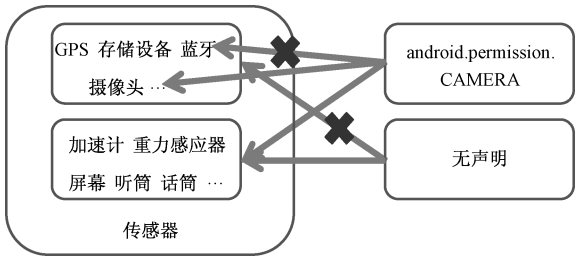

图 3　Android 权限控制机制

Fig. 3　Android permission mechanism

3.2　Android 传感器隐蔽信道原理

根据形成机理的不同,Android 系统隐蔽信道分为基于共享资源的 Android 隐蔽信道和基于传感器设备的 Android 隐蔽信道. 基于传感器设备的隐蔽信道只需利用一个应用程序即可实现用户数据的泄漏.

定义 2　(Android 传感器隐蔽信道, Android Covert Channel based Sensors) 基于传感器的 Android 隐蔽信道可用如下 5 元组及描述公式进行表述:

$< Perm, APPs, Sensors, Confidential, \rightarrow >$
If $\exists a \in APPs, \exists s \in Sensors, \exists c \in Confidential$
And $Perm(a,s) = \Theta \wedge (a \rightarrow s) = True$
Then $(a \rightarrow c) = True$

其中,Android 系统中未进行权限保护的传感器 s,任意应用程序都可以直接进行访问,如果应用程

序 a 能够访问 s,则 $Perm(a,s) = \Theta \wedge (a \rightarrow s) = True$;应用程序 a 实时监控传感器 s 的采集信息,并根据用户的行为进行数据挖掘,从而能够推断用户的机密信息,最终导致用户机密信息泄漏,即 $(a \rightarrow c) = True$.

例如,恶意应用程序通过监控重力加速计的实时数值,从而推断用户在屏幕上的输入字母,最终窃取用户输入的密码信息.

3.3 Android 传感器隐蔽信道限制原理

任意一款应用程序对底层传感器的访问需要经过 Android Application 层、Framework 层、Library 层、Linux kernel 层和硬件层,如图 4 所示.

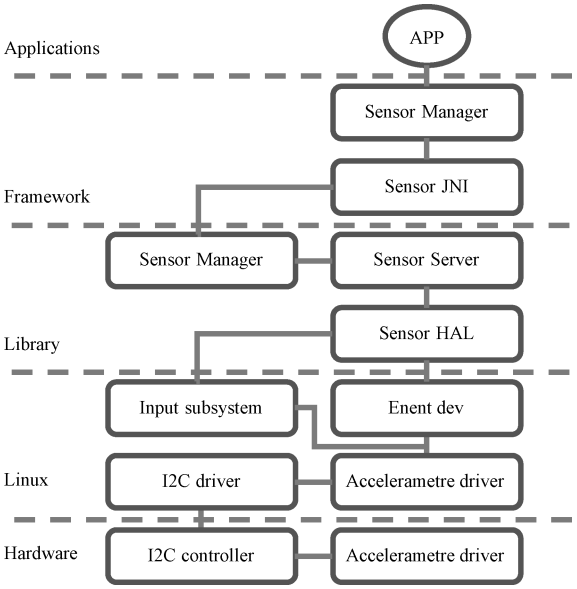

图 4 Android 系统传感器控制架构

Fig. 4 Control structure of Android sensors

Application 层即具体的应用程序,用来接收 Sensor 返回的数据,并处理实现对应的 UI 效果,如屏幕旋转、打电话时灭屏、自动调接背光等;Framework 层的 JNI 负责访问 Sensor 的客户端;Library 是动态库,封装了整个 Sensor 的 IPC 机制,如 SensorManager 是客户端,SensorService 是服务端,而 HAL 部分是封装了服务端对 Kernel 的直接访问;在 Linux kernel 层,Sensor 驱动要注册到 Input Subsystem 上,然后通过 Event Device 把 Sensor 数据传到 HAL 层;最后 Sensor 硬件要挂在 I2C 总线上.

限制基于传感器的 Android 系统隐蔽信道,需要对未受保护的 Sensor 提供完整的权限机制.

定义 3 (Android 传感器隐蔽信道限制方案) 基于传感器的 Android 隐蔽信道可用如下 5 元组 $< Perm, APPs, Sensors, Confidential, \rightarrow >$ 及描述公式进行表述:

$\exists a \in APPs, \exists s \in Sensors, \exists c \in Confidential$

If $Perm(a,s) = \Theta \wedge (a \rightarrow s) = True \wedge (a \rightarrow c) = True$ Then

$\exists Perm, \forall a \in APPs, Perm(a,s) \neq \Theta$

其中,如果应用程序 a 能够无限制地访问传感器 s,即 $Perm(a,s) = \Theta \wedge (a \rightarrow s) = True$;并通过对传感器数据的分析推断用户输入的机密信息,即 $(a \rightarrow c) = True$;则需要对该传感器 s 进行权限限制,即 $\exists Perm, \forall a \in APPs, Perm(a,s) \neq \Theta$,从而限制该基于传感器 s 的 Android 系统隐蔽信道.

4 实验分析

本节通过对 Framework 层源代码的修改,在 Android 系统中实现针对重力加速计传感器的权限控制,并用实际的实验分析展示基于权限控制机制的 Android 传感器隐蔽信道的限制效果.

4.1 Android Framework 层添加传感器权限控制

frameworks/base/core/res/AndroidManifest. xml 定义所有 Android 的权限,在该文件中添加准备定义的权限,最终该文件会生成 Android 权限文件 out/target/common/R/android/Manifest. java.

```
#文件位置 frameworks/base/core/res/AndroidManifest. xml
#定义权限 android. permission. SENSOR_ENABLE
<! — Allows an application to sensor_enable demo — >
<permission
    android:name = "android. permission. SENSOR_ENABLE"
    android: permissionGroup = " android. permission-group.
HARDWARE_CONTROLS"
    android:protectionLevel = "dangerous"
    android:label = "@ string/permlab_sensor_enable"
    android:description = "@ string/permdesc_sensor_enable"
/ >
#定义权限 android. permission. SENSOR_INFO
<! — Allows an application to sensor_info demo — >
<permission
    android:name = "android. permission. SENSOR_INFO"
    android: permissionGroup = " android. permission-group.
HARDWARE_CONTROLS"
    android:protectionLevel = "normal"
    android:label = "@ string/permlab_sensor_info"
    android:description = "@ string/permdesc_enable"
/ >
```

frameworks/base/core/res/res/values/strings.xml 是权限的文字描述部分,在每个 Android 应用程序的设置文件中会显示这个内容.

```
#位置 frameworks/base/core/res/res/values/strings.xml
<! — Title of an application permission, listed so the user can choose whether they want to allow the application to do this. — >
<stringname = “permlab_sensor_enable” > sensor enable </string >
<! — Title of an application permission, listed so the user can choose whether they want to allow the application to do this. — >
<stringname = “permlab_sensor_info” > sensor info </string >
```

frameworks/base/core/java/android/app/ContextImpl.java 中的 registerService 用来注册 Sensor 服务,用户将字符串 SENSOR_SERVICE 和真正的服务绑定. 未修改的 SensorManager 只有一个参数,此处添加一个参数 ctx,以便后续用于检查权限.

```
#文件位置
frameworks/base/core/java/android/app/ContextImpl.java
registerService(SENSOR_SERVICE, new ServiceFetcher() {
public Object createService(ContextImpl ctx) {
    return new SensorManager(ctx,ctx.mMainThread.
    getHandler().getLooper());
}});
```

frameworks/base/core/java/android/hardware/SensorManager.java 是权限判断的关键部分. 其逻辑为:如果有权限 android.permission.SENSOR_ENABLE,则判断权限 android.permission.SENSOR_INFO;如果有 SENSOR_INFO 权限,权限控制机制将允许 Sensor 操作;若没有 android.permission.SENSOR_ENABLE 权限,权限控制机制虽然允许对 Sensor 操作,但是不允许访问其数据. 这样做的目的是,为了和现有程序兼容,达到实验验证效果.

```
#文件位置
frameworks/base/core/java/android/hardware/SensorManager.java
public SensorManager(Context context,Looper mainLooper) {
        int flag = 0;
        mMainLooper = mainLooper;
        Log.e(TAG," checking
        android.permission.SENSOR_ENABLE");
        if (context.checkCallingOrSelfPermission(" android.permission.
SENSOR_ENABLE") = = PackageManager.PERMISSION_GRANTED){
            Log.e(TAG,“permission SENSOR_ENABLE”);
            flag = 1;
        }
        Log.e(TAG," checking
        android.permission.SENSOR_INFO");
        if (    (flag = = 1) &&
        (context.checkCallingOrSelfPermission(" android.permission.
SENSOR_INFO")
        ! = PackageManager.PERMISSION_GRANTED)) {
Log.e(TAG," demo no android.permission.SENSOR_INFO");
throw new SecurityException (“requires SENSOR_INFO
permission”);
}
```

4.2　实验结果分析

首先实现基于传感器 Android 系统隐蔽信道,并验证在未加入权限控制的 Android 系统中,该隐蔽信道能够实现机密信息泄漏.

如图 5 所示,该应用程序能够访问 Android 重力加速计数据,并进行该数据的发送;实验中设置了一台 Web Server 用于接收该应用程序发送的传感器数据(如图 6 所示);在服务器上,恶意用户通过对传感器数据的分析,从而推断用户在 Android 系统上的操作,最终得到用户的输入密码等机密信息.

图 5　Android 应用程序通过隐蔽信道获得和传输传感器数据

Fig. 5　Sensor data got and sent by Apps via covert chanel

```
{'sensor': u'0: 322.9258, -33.12742, -1.1824392;A: -0.20731887, 5.228627, 7.993078'}
{'sensor': u'0: 322.9258, -33.12742, -1.1824392;A: -0.20731887, 5.228627, 7.993078'}
{'sensor': u'0: 322.9258, -33.12742, -1.1824392;A: -0.20731887, 5.228627, 7.993078'}
```

图 6　恶意服务通过 Android 隐蔽信道获得传感器数据

Fig. 6　Android sensor data got by malicious server via covert channel

其次,在 Android 系统中添加针对 Sensor 传

感器即重力加速计的权限控制机制. 重新实现 Android 应用程序,在 AndroidManifest. xml 中添加两个权限,如下所示:

```
#权限声明
< uses-permission
    android:name = " android. permission. SENSOR_INFO" / >
< uses-permission
    android:name = " android. permission. SENSOR_ENABLE" / >
```

编译打包应用程序源代码,并在 Android 模拟器中进行安装,运行效果如图 7 所示. 证明可以获得 SENSOR_INFO 权限,应用程序程序能正确执行,不影响用户的正常使用.

图 7 声明传感器权限的 Android 程序运行结果

Fig. 7 Result of sensor using the right permission of Android APP

再次实现 Android 程序,在 AndroidManifest. xml 中只添加 android. permission. SENSOR _ ENABLE 权限,取消 android. permission. SENSOR_ INFO 权限. 重新编译打包,并在 Android 模拟器中安装执行,运行效果如图 8 所示. 证明没有 SENSOR_INFO 权限,程序将不能正确执行,从而能够限制基于传感器的 Android 系统隐蔽信道.

图 8 未声明传感器权限的 Android 程序运行结果

Fig. 8 Result of sensor without the right permission of Android APP

5 总结

隐蔽信道研究是信息安全的一个重要难题,移动智能终端隐蔽信道是 IT 产业发展中的全新问题. 移动智能终端涉及大量的用户隐私数据,包括用户通讯录信息、短信信息、网上银行信息、地理位置信息、用户日程、用户行为习惯等,任何系统缺陷及安全隐患都会影响数以 10 亿计的用户. Android 系统隐蔽信道比经典的攻击行为更为复杂,在用户无察觉的情况下实现隐私窃取和泄漏,并且目前仍缺乏有效的消除限制方法. 本文将 Android 系统隐蔽信道细分为基于共享资源和基于传感器的隐蔽信道两种基本模型,并深入研究传感器隐蔽信道的形成机理;通过对 Android 系统权限控制安全机制的分析,扩展了权限控制机制的保护范围;设计和实现了基于权限控制机制的 Android 系统传感器隐蔽信道限制方法;并通过实验证明该方法在实际的隐蔽信道限制中能够达到切实可用的限制效果;该方法可以向更多类型的传感器进一步扩展,从而实现对移动智能终端的安全防护.

参考文献

[1] Zhou Y J, Jiang X X. Detecting passive content leaks and pollution in Android applications [C] // Proc of the 20th Annual Network and Distributed System Security Symposium (NDSS'13). San Diego, California, USA, 2013: 1-16.

[2] Grace M, Zhou Y J, 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 (MobiSys'12). Low Wood Bay, Lake District, UK, 2012: 281-294.

[3] La Polla M, Martinelli F, Sgandurra D. A survey on security for mobile devices [J]. IEEE Communications Surveys & Tutorials, 2012, 15(1): 446-471.

[4] Wu J Z, Wu Y J, Yang M T, et at. Vulnerability detection of Android system in fuzzing cloud [C] // Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing. IEEE Computer Society, 2013: 954-955.

[5] Enck W, Octeau D, Mcdaniel P, et al. A study of android application security [C] // Proceedings of the 20th USENIX Conference on Security. San Francisco, California, USA: USENIX Association,2011: 21-36.

[6] Schlegel R, Zhang K H, Zhou X Y, et al. Soundcomber: a stealthy and context-aware sound trojan for smartphones [C] // Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS'11). San Diego, California, USA,2011: 17-33.

[7] Lampson B W. A note on the confinement problem [J]. Commun ACM, 1973, 16(10): 613-615.

[8] 王永吉, 吴敬征, 曾海涛,等. 隐蔽信道研究 [J]. 软件学报, 2010, 21(9): 2 262-2 288.

[9] Cai L, Chen H. TouchLogger: inferring keystrokes on touch screen from smartphone motion [C] // Proceedings of the 6th USENIX Conference on Hot Topics in Security. San Francisco, California, USA: USENIX Association. 2011,9-14.

[10] Wu J Z, Wu Y J, Wu Z F, et al. Vulcloud: scalable and hybrid vulnerability detection in cloud computing [C] // Proceedings of the Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on. 2013: 18-20.

[11] Yang M T, Wu J Z, Wu Y J, et al. Policykeeper: recommending proper security mechanisms based on the severity of vulnerability considering user experience [C] // Proceedings of the Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on. 2013:F 18-20.

[12] Au K W Y, Zhou Y F, Huang Z, et al. PScout: analyzing the Android permission specification [C] // Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS'12). Raleigh, North Carolina, USA, 2012: 217-228.

[13] Shah G, Molina A, Blaze M. Keyboards and covert channels [C] // Proceedings of the 15th Conference on USENIX Security Symposium: Volume 15. Vancouver B C, Canada: USENIX Association,2006.

[14] Owusu E, Han J, Das S, et al. ACCessory: password inference using accelerometers on smartphones [C] // Proceedings of the 12th Workshop on Mobile Computing Systems & Applications (HotMobile'12). San Diego, California, USA, 2012: 1-6.

[15] Miluzzo E, Varshavsky A, Balakrishnan S, et al. Tapprints: your finger taps have fingerprints [C] // Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12). Low Wood Bay, Lake District, UK, 2012: 323-336.

[16] 王昌达, 鞠时光, 周从华,等. 一种隐通道威胁审计的度量方法 [J]. 计算机学报, 2009, 32(4): 751-762.

[17] 卿斯汉, 朱继锋. 安胜安全操作系统的隐蔽通道分析 [J]. 软件学报, 2004, 15(09): 1 385-1 392.

[18] Wu J Z, Wang Y J, Ding L P, et al. A practical covert channel identification approach in source code based on directed information flow graph [C] // The Fifth Annual International Conference on Secure Software Integration and Reliability Improvement (SSIRI 2011). Jeju Island, Korea, Jun 27-29, 2011: 98-107.

[19] Wu J Z, Wang Y J, Ding L P, et al. Constructing scenario of event-flag covert channel in secure operating system [C] // 2nd International Conference on Information and Multimedia Technology (ICIMT 2010). Hongkong, Dec 28-30, 2010: 371-375.

[20] Yao L H, Zi X C, Pan L, et al . A study of on/off timing channel based on packet delay distribution[J] Compwters & Security, 2009, 28(8):785-794.

[21] Wu J Z, Wang Y J, Ding L P, et al. Improving performance of network covert timing channel through Huffman coding [J]. Mathematical and Computer Modelling, 2012, 55(1/2): 69-79.

[22] 王永吉, 吴敬征, 曾海涛,等. 一种基于并发冲突间隔时间的隐蔽信道检测方法 [J]. 计算机研究与发展, 2011, 48(8): 1 542-1 553.

[23] Wu J Z, Wang Y J, Ding L P, et al. Identification and evaluation of sharing memory covert timing channel in Xen virtual machines [C] // IEEE 4th International Conference on Cloud Computing (CLOUD 2011). Washington DC, USA, Jul 4-9, 2011: 283-291.

[24] Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds [C] // Proceedings of the 16th ACM Conference on Computer and Communications Security. Chicago, Illinois, USA: ACM, 2009: 199-212.

[25] 吴敬征, 丁丽萍, 王永吉. 云计算环境下隐蔽信道关键问题研究 [J]. 通信学报, 2011, 32(9A): 184-203.

[26] Sailer R, Jaeger T, Valdez E, et al. Building a MAC-based security architecture for the Xen open-source hypervisor [C] // Proceedings of the Computer Security Applications Conference, 21st Annual. F, 2005.

[27] Payne B D, Carbone M, Sharif M, et al. Lares: an architecture for secure active monitoring using virtualization [C] // Proceedings of the Security and Privacy, 2008 SP 2008 IEEE Symposium on. F, 2008.

[28] Azab A M, Ning P, Wang Z, et al. HyperSentry: enabling stealthy in-context measurement of hypervisor integrity [C] // Proceedings of the 17th ACM Conference on Computer and Communications Security. Chicago, Illinois, USA: ACM, 2010: 38-49.

[29] Wang Z, Jiang X X. HyperSafe: a lightweight approach to provide lifetime hypervisor control-flow Integrity [C] // Proceedings of the Security and Privacy (SP), 2010 IEEE Symposium on. F 16-19 May, 2010.

[30] Seshadri A, Luk M, Qu N, et al. SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes [J]. SIGOPS Oper Syst Rev, 2007, 41(6): 335-350.

[31] Wu J Z, Ding L P, Lin Y Q, et al. XenPump: a new method to mitigate timing channel in cloud computing [C] // Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing (CLOUD'12). Hawaii, USA: IEEE Computer Society,2012: 678-685.

[32] Tsai C R. Covert channel analysis in secure compwter systems [D]. University of Maryland, College Park, MD,1987.