第44卷第10期 中国科学技术大学学报 Vol. 44, No. 10
2014年10月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Oct. 2014

文章编号:0253-2778(2014)10-0853-09

一种针对 Android 系统隐私保护机制有效性的评估方法

曾述可[1],张 阳[2],程 亮[2],邓 艺[2],冯登国[1,2]

(1. 中国科学技术大学计算机科学与技术学院,安徽合肥 230027;2. 中国科学院软件研究所,北京 100190)

摘要:为了保护用户的隐私数据,Android 实现了一套基于权限的安全机制. 为此设计了一种针对该机制的评估工具 PrivacyMiner,以检测其在隐私保护方面的有效性. 首先将 Android 系统中的隐私数据分为 22 个类别;然后使用动态检测与静态污点分析相结合的方法,来检测 Android 系统的安全机制是否能有效地保护它们. 用 PrivacyMiner 工具对 12 个版本的 Android 源代码进行了检测,发现其中有 7 个类别的隐私数据并没有得到有效的保护,恶意软件可以在用户不知情的情况下读取这些隐私,并发送到任意服务器上. 这些漏洞在 6 个 Android 设备上得到了验证,从 Android 2.1 到最新发布的 Android 4.4.2,均得到了 Android 安全团队的确认.

关键词:Android;隐私保护;验证;污点分析;静态分析

中图分类号:TP393 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2014.10.009

引用格式:Zeng Shuke, Zhang Yang, Cheng Liang, et al. An approach to evaluate the effectiveness of privacy protection in Android system[J]. Journal of University of Science and Technology of China, 2014, 44(10):853-861.

曾述可,张阳,程亮,等. 一种针对 Android 系统隐私保护机制有效性的评估方法[J]. 中国科学技术大学学报,2014,44(10):853-861.

An approach to evaluate the effectiveness of privacy protection in Android system

ZENG Shuke[1], ZHANG Yang[2], CHENG Liang[2], DENG Yi[2], FENG Dengguo[1,2]

(1. *School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027, *China*; 2. *Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: To protect private data in smart phones, Android enforces a permission-based security policy. PrivacyMiner, a tool for evaluating the effectiveness of privacy protection in Android, was designed. First, 22 categories of private data in smart phones were identified, which were then checked to see if Android could efficiently protect them from malware. PrivacyMiner was applied to 12 revisions of Android source code, and it was found that 7 categories of private data were not well protected, as Malware can read them and send them out without any permission. These vulnerabilities were verified on 6 Android devices with 6 revisions of Android, from 2.1 up to 4.4.2. Our findings were confirmed by the Android Security Team from Google.

Key words: Android; privacy protection; evaluation; taint analysis; static analysis

收稿日期:2014-01-17;**修回日期**:2014-03-13

基金项目:国家自然科学青年基金(61100227),中国高技术研究发展(863)计划(2011AA01A203)资助.

作者简介:曾述可,男,1980 年生,博士生. 研究方向:操作系统安全. E-mail: skzeng@mail.ustc.edu.cn

通讯作者:冯登国,研究员/博士生导师. E-mail: feng@tca.iscas.ac.cn

0 引言

当前智能手机在移动终端市场具有很高的占有率.智能手机用户可以下载并安装第三方软件,来扩展手机的功能.苹果公司的App Store市场已有超过500亿次[1]的软件下载,而Google公司的Google Play也有超过480亿次[2]的下载.这些第三方的软件会访问手机的通讯模块、存储器以及传感器模块,因此可能会产生隐私泄漏问题.例如,如果一个软件既访问手机里的通话记录,又可以通过网络向外发送数据,那么它就有可能把手机里的通话记录发送给外部服务器.

作为当前市场占有率最高的智能手机操作系统,Android实现了一套基于权限管理的安全机制,在该机制中,应用软件需要声明相关的权限,才能访问特定的功能[3-5].在Android系统中,应用软件如需读取通话记录,就要声明"READ_CALL_LOG"权限,如需通过网络发送数据,就要声明"Internet"权限.用户在Google Play下载并安装一个软件之前,会收到软件相关权限的提示,用户则根据该提示,决定是否继续安装.

基于权限管理的安全机制,是Android隐私保护的基础,近年来受到学术界和产业界的广泛关注[6-8].根据Android系统的设计思想,用户负责根据软件声明的权限来决定是否安装一个软件,Android系统负责保证一个软件不能访问无权访问的数据或功能.当前的多项研究[9-10]表明,用户难以理解Android系统的各种权限,许多用户在安装软件之前,不会留意相关的权限提示.另一方面,如果手机开发商或应用软件开发者不能很好地遵循Android的权限规范,会让恶意软件能够绕过权限检查机制[11-14];即使用户很好地理解了Android的权限机制,并且手机开发商和应用软件开发者完全遵循了Android的权限规范,Android的权限管理机制仍然不能有效地保护用户的隐私数据.

本文设计了一种基于动态检测和静态污点分析的检查工具PrivacyMiner.该工具分两个步骤对Android的隐私保护机制进行检测,首先将Android设备上的隐私数据划分为22个类别,检测Android的隐私保护机制能否有效地防止这些隐私数据被恶意软件读取,然后检查恶意软件能否将读取到的隐私数据发送出去,而无需声明任何相应的权限.

我们使用PrivacyMiner来检测了12个版本的Android源代码,从2.1到最新的4.4.2.我们发现Android的安全机制并不能保护所有的隐私数据,在22类隐私数据中,有7个类别没有得到有效的保护.此外,我们还发现了一条可能被恶意软件利用的执行路径,通过该路径,恶意软件可将用户隐私通过网络发送到外部服务器上,而无需声明相应的权限.我们在6个Android设备上对这些隐私问题进行了验证,结果表明,这些隐私问题已经存在多年,我们的结果得到了Android安全团队的确认.

1 相关工作

1.1 Android安全机制

研究表明,用户难以理解权限管理系统,用户在安装一个应用软件之前,经常不会留意该软件所声明的权限[9-10,15-16].随着Android系统的发展,权限系统也在不断变化,新的权限被不断添加到Android系统中,这让用户越来越难以理解[15,17].此外,应用软件还存在重新打包的问题[18-19],恶意软件在正常软件中添加恶意代码,重新打包后发布在软件市场上,这让用户更加难以识别.

此外,应用软件的开发人员经常不能很好地遵循Android的权限规范[8],他们开发的软件往往也为恶意软件提供了便利:恶意软件可以调用这些软件的相关组件读取或发送隐私.Chin等[11-12]提出了ComDroid工具,用于分析Android系统中的应用软件之间的通信所带来的风险.他们分析了20个第三方软件,并发现其中至少有12个有安全漏洞.Grace等[13]提出了Woodpecker工具,以检查手机制造商是否遵循了Android权限管理的规范.Felt等[20]提出了Stowawary工具,用于检测第三方软件在开发过程中是否遵循了最小权限的原则:应用软件如果用不到某项权限,就不应该声明它.Chan等[21]提出了DroidChecker工具,用于检查第三方软件是否存在隐私泄露的问题.

与现有工作不同,本文的工作揭示了Android系统中更加严重的问题:即使用户很好地理解了Android的权限管理规范,并且手机厂商和第三方应用软件的开发者严格遵循了这一规范,Android系统仍然不能有效地阻止隐私泄漏.

1.2 污点分析

当前有许多污点分析方面的研究[22-26],致力检查Android系统中第三方软件的隐私泄漏问题,这些研究工作使用静态或动态污点分析的方法对第三

方应用软件进行检测，能够识别一些恶意软件，这些恶意软件侵犯了用户的隐私，包括 IMEI、地理数据、短信等. 虽然在这些工具都能找出一些存在隐私问题的第三方软件，但它们都存在一些不足. 例如，这些侵犯用户隐私的第三方软件，都声明了读取隐私的权限，用户会在安装这些软件之前，得到警告. 本文的研究表明，恶意软件可以读取并发送多种类型用户的隐私数据，而不需要声明相应的权限；并且，当前多研究工作[22,24]都过于依赖应用软件的权限声明，如果恶意软件没有在该文件中声明与隐私相关的权限，就能绕过这些工具的检测.

2　Android 权限管理系统

2.1　Android 的权限管理

为了保护用户的隐私数据，Android 实现了一套基于权限的安全机制：敏感的数据和资源，都由相关的权限进行保护. 应用软件如需访问这些数据或资源，必须声明相关的权限.

Android 权限管理系统的责任分为三个部分：应用软件开发者、用户、Android 系统. 应用软件开发者应当声明该软件所需要的权限；用户在安装该软件前，Google Play 会提示这个软件声明了哪些权限，用户再根据这些提示决定是否继续安装这个软件；在用户运行该软件时，Android 系统负责保证该软件不能执行声明权限之外的操作，这一功能由 Application Framework 中相关的 API 函数以及内核中的 Binder 驱动共同完成.

2.2　组件间通信

在 Android 系统中，同一应用软件的组件之间，或不同应用软件的组件之间，可以进行通信，这种组件间通信的机制为恶意软件提供了便利.

Android 应用软件通常由四种类型的组件构成：Activity 用于展现可视化的用户界面；Service 是后台运行的服务进程；Broadcast Receiver 用于接收来自 Android 系统或其他组件的广播；Content Provider 用于对外提供私有数据的读写接口. 通常一个 Android 应用软件由多个这样的组件构成，为了各组件之间能够传递数据，Android 实现了一套组件间通信的方法.

Android 的组件间通信机制，使开发者可以调用系统功能或其他应用软件，但是，这种机制也会导致 Android 系统中权限泄露. 例如，如果一个应用软件没有声明“READ_SMS”权限，就不能读取手机中的短信，但是它可以调用其他有权限的组件读取短信，并把数据传递给它.

在 Android 系统中，组件间通信的接口，其作用相当于 API 函数调用. 例如，应用软件需要调用原生软件 Telephony 中的一个 Activity 组件，来实现拨打电话的功能，如果原生软件中存在安全漏洞，就有可能被恶意软件利用，这些漏洞将影响所有安装了该原生软件的 Android 设备.

2.3　攻击模型

如果一个应用软件可以在用户不知情的情况下窃取用户的隐私数据，那么我们就认为这个软件是一个恶意软件，并且 Android 的权限管理机制受到了损害. 由于用户在安装一个应用软件之前，Google Play 会提示用户这个软件声明了哪些权限，用户再决定是否安装，所以在本文的攻击模型中，恶意软件不应该声明任何读取隐私数据的权限.

由于 Google Play 中存在许多第三方应用软件，它们具有读取隐私数据的权限，但又没有严格遵循 Android 的权限规范，容易被恶意软件用于获取隐私数据. 如果一个恶意软件利用第三方软件读取或发送隐私数据，那么它在使用上就具有一定的局限性——只有安装特定第三方软件的设备，才会受到恶意软件的攻击. 在本文的攻击模型中，假定恶意软件无法利用任何第三方软件.

当恶意软件成功读取了用户的隐私数据后，它们需要把这些隐私数据发送出去. 隐私的发送过程不能被用户察觉到，因此恶意软件不能声明任何相关的权限，例如拨打电话、发送短信、使用网络等.

3　PrivacyMiner 的设计与实现

本文设计并实现了 PrivacyMiner 工具，用于检测 Android 系统的隐私保护机制. PrivacyMiner 的系统设计如图 1 所示，它包含一个动态检查模块和一个静态检查分析模块.

3.1　动态检查模块

PrivacyMiner 工具包含一个动态检查模块，它是一个运行在 Android 模拟器或设备中的应用软件，用于测试隐私数据或者相关模块是否可以被恶意软件调用，而无需声明相关的权限. 如恶意软件是否能够读取手机中的短信，而无需声明“READ_SMS”权限. 动态检查模块还要检查隐私相关的文件或传感器是否可以被恶意软件直接读取，如保存联系人和短信的 sqlite 数据库文件等.

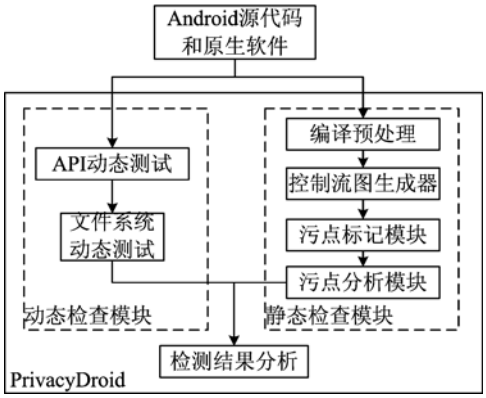

图 1 PrivacyMiner 系统设计

Fig. 1 System overview of PrivacyMiner

3.2 静态检查模块

PrivacyMiner 的静态检查模块检测的对象是 Android 系统的 Application Framework 和原生的应用软件. 原理是首先生成它们的控制流图,然后再在控制流图的基础上进行污点标记和分析.

3.2.1 静态控制流图生成器

在将 Application Framework 和原生应用软件编译为字节码后,控制流图生成器为每一个待分析的类生成一个控制流图. 在控制流图中,源代码中的每一个方法被视为一个基本块,每个方法的入口被视为基本块的入口;每一条返回语句或者异常处理语句都被视为基本块的结束点. 一个控制流图通常是个有向图:图中的结点代表代码中的基本块;图中的边代表程序中的方法调用或者跳转语句.

与一般的 Java 程序控制流图不同,在生成控制流图时,还需要考虑两个额外的因素:一个是 Android 系统提供的组件间通信机制,另一个是 Application Framework 中的反射机制.

对于 Android 系统的组件间通信机制,我们先分析所有发送端的 API 函数,如 sendBroadcast()、startActivity() 等以及所有作为接收端的 BroadcastReceiver 和 IntentFilter;然后,匹配发送端和接收端,将一对发送端和接收端视为控制流图中的一条有向边. 如果一个应用程序通过 startActivity() 函数发送"android. intent. action. CALL"类型的数据,Android 系统将会把这些数据转发给在接收端 IntentFilter 中注册这个类型的组件,于是在控制流图中从 startActivity() 到 IntentFilter 间就生成一条有向边.

由于 Java 反射机制的存在,恶意软件能够访问 Android 的 API 函数. 通常情况下这些隐藏的函数只供 Android 系统调用,不提供给第三方的应用软件. 为了跟踪恶意软件对这些隐藏函数可能的调用,PrivacyMiner 列出所有 Application Framework 中所有可以被反射机制访问的隐藏 API 函数,并在分析过程中将它们视为正常的 API 函数.

由 PrivacyMiner 生成的静态控制如图 2 所示. 其中控制流图的结点编号,对应于源代码中的行号.

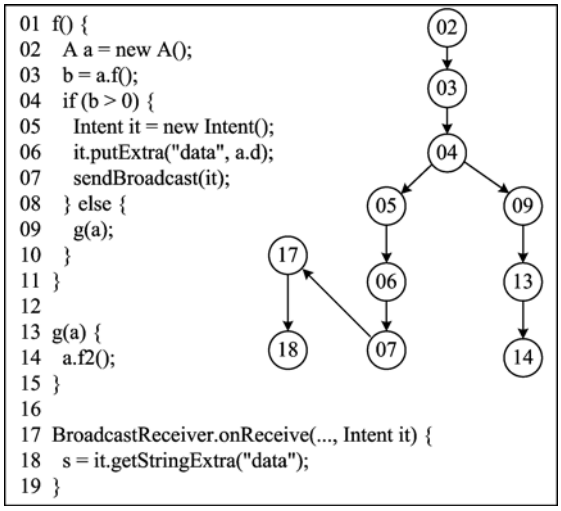

图 2 静态控制流图

Fig. 2 Static control-flow graph

3.2.2 污点标记模块

在生成静态控制流图后,PrivacyMiner 在控制流图中标记污点源和汇聚点,再检查从污点源到汇聚点之间,是否存在可能的执行路径.

PrivacyMiner 的污点标记,分为两个步骤:私隐读取和隐私发送.

对于隐私读取,PrivacyMiner 将隐私数据标记为污点源,并将所有可向应用软件传递隐私数据的接口,标记为汇聚点. PrivacyMiner 将所有的隐私数据划分为 22 个类别,如表 1 所示. 这些数据都被标记为污点源. 汇聚点包括所有可以读取隐私数据中 API 或组件间通信接口,例如 ContentResolver. query()、sendBroadcast()等.

对于隐私发送,PrivacyMiner 将所有能够传递给 Application Framework 或原生软件的数据标记为污点源,包括所有能够向 Application Frame 传递参数的 API 函数以及原生软件中所有组件间通信的接口,如 IntentFilter,onCreate()等. 汇聚点包括所有向外发送数据的接口,如发送短信、拨打电话以及通过网络发送数据的接口.

表 1　Android 系统的隐私类型与保护机制

Tab. 1　Categories of privacy & protection

隐私类别	相关保护机制
应用软件数据	文件或权限管理系统的相关权限
摄像头	CAMERA 权限
麦克风	RECORD_AUDIO 权限
联系人	READ_CONTACTS 权限
通话记录	READ_CONTACTS 或 READ_CALL_LOG 权限
短信	READ_SMS 等权限
屏幕内容	READ_FRAME_BUFFER 权限
系统日志	READ_LOGS 或 DUMP 权限
系统状态	READ_PHONE_STATE 等权限
日程	READ_CALENDAR 权限
用户配置	READ_PROFILE 等权限
用户账户	GET_ACCOUNTS 权限
NFC 数据	NFC 权限
外部存储器	READ_EXTERNAL_STORAGE 权限
浏览器书签	READ_HISTORY_BOOKMARKS 权限
浏览器历史	READ_HISTORY_BOOKMARKS 权限
软件运行状态	GET_TASKS 等权限
系统内核状态	Linux 内核的访问控制机制
用户位置信息	ACCESS_COARSE_LOCATION 等权限
隐私传感器	各种相关权限
应用软件信息	无
剪贴板	无

3.2.3　污点分析模块

在静态控制流图中标记了污点源和汇聚点后，PrivacyMiner 将检测污点源与汇聚点之间，是否存在满足特定条件的执行路径，使得污点数据能够从污点源传播到汇聚点. 为了追踪污点数据的传播，PrivacyMiner 实现了一个污点分析模块.

由于控制流图生成器已将 Android 的组件间调用和由 Java 反射机制调用的 API 函数映射成普通的函数调用，因此污点分析模块只需要跟踪控制流图中的两类语句：赋值语句和函数调用.

我们为每一个操作数维护一个布尔标签 t，用于表示该操作数是否被标记为污点. 为了能更加精确地跟踪污点数据的传播，我们为类实例中的每一个成员变量单独维护一个污点标签：假设 d 是类 D 的一个实例，它有两个成员变量 a 和 b，那么我们分别为它们维护标签 d. a. t 和 d. b. t.

令 C 为常量，a、b、c 为变量，$\oplus$ 为一元或多元运算符，$f(a_1, a_2, \cdots, a_n)$ 为有 n 个参数的函数调用，其返回值为 fr，函数执行过程中可能抛出的异常为 fe，p 为访问隐私数据和功能模块的 API 函数. 我们的污点传播规则如表 2 所示.

表 2　污点传播规则

Tab. 2　Rules of taint propagation

语句	污点传播规则
$a=C$	a. t=False
$a=p(\cdots)$	a. t=True
$a=b$	a. t=b. t
$a=\oplus\ b$	a. t=b. t
$a=b\oplus C$	a. t=b. t
$a=b\oplus c$	a. t=b. t $\cup$ c. t
$a=f(a_1,\cdots,a_n)$	a. t=fr. t $\cup$ fe. t

对于赋值语句，如果任何一个源操作数被标记为污点，那么目的操作数也将被标记为污点. 对于函数调用，如果函数参数被标记为污点，那么这些污点信息会继续在函数执行路径上进行传播；函数在执行过程中也会引入新的污点. 例如，执行了某条读取短信的语句，那么这条语句的目的操作数将被标记为新的污点. 如果函数的返回值或执行过程中抛出的异常被标记为污点，那么调用函数时的目的操作数也将被标记为污点.

对于一条泄漏通话记录的执行路径，其污点分析过程如表 3 所示.

表 3　污点标记与传播规则示例

Tab. 3　Example of taint propagation

源代码	污点标记与传播
ContentResolver cr=getContentResolver();	cr. t=False
Cursor c = cr. query (CallLog. Calls. CONTENT_URI, …);	c. t=True
int i = c. getColumnIndex (CallLog. Calls. NUMBER);	i. t=True
String address=c. getString(i);	address. t=True
Intent it=new Intent();	it. t=False
it. putExtra("Address", address);	it. t=True
sendBroadcast(it);	汇聚点

如果污点分析模块发现了一条从污点源到汇聚点的执行路径，那么我们认为这是一条可能造成隐私泄露的执行路径，我们再人工确认这条路径是否可以被恶意软件利用.

我们的污点传播规则构成了一个格结构. 令 L 为控制流图中所有操作数的集合，操作数的污点有如下 4 种取值：$\perp$、untainted、tainted、T. 其中，$\perp$ 表示未确定操作数是否有污点；untainted 表示操作数被标记为非污点；tainted 表示操作数被标记为污点；T 表示汇聚点. 我们定义格上的二元关系 $\leqslant$ 如下：

$$\perp \leqslant \text{untainted} \leqslant \text{tainted} \leqslant T.$$

令 x、y、z 为操作数的污点取值，则该二元关系满足如下性质：

自反：$x \leqslant x$；

反对称：如果 $x \leqslant y$，$y \leqslant x$，则 $x = y$；

传递：如果 $x \leqslant y$，$y \leqslant z$，则 $x \leqslant z$.

因此 $\leqslant$ 是偏序关系.

定义 L 上的两个二元运算 $\cap$，$\cup$ 如下：

$$x \cap y = \begin{cases} \text{tainted}, & x \leqslant \text{tainted and } y \leqslant \text{tainted} \\ \text{untainted}, & \text{otherwise} \end{cases},$$

$$x \cup y = \begin{cases} \text{tainted}, & x \leqslant \text{tainted or } y \leqslant \text{tainted} \\ \text{untainted}, & \text{otherwise} \end{cases}.$$

则这两个二元运算满足如下性质：

交换率：$x \cap y = y \cap x$

$x \cup y = y \cup x$

结合率：$x \cap (y \cap z) = (x \cap y) \cap z$

$x \cup (y \cup z) = (x \cup y) \cap z$

吸收律：$x \cup (x \cap y) = x$

$x \cap (x \cup y) = x$

于是 $<L, \leqslant, \cap, \cup>$ 构成了一个格结构，因此，我们的污点传播规则是收敛的，控制流图中每一个操作数最终都将被赋予一个确定的污点状态.

4 PrivacyMiner 的检测方法

由于每一个隐私泄漏都包含两个步骤：读取隐私和发送隐私，因此 PrivacyMiner 也分两个步骤进行检测：①恶意软件是否能读取这些隐私，而无需声明相应的权限；②恶意软件在读取到隐私数据后，能否将它们发送出去，而无需声明相应的权限.

4.1 隐私读取

PrivacyMiner 检测隐私读取的过程如图 3 所示，它对所有 22 个类别的隐私数据分别进行检测.

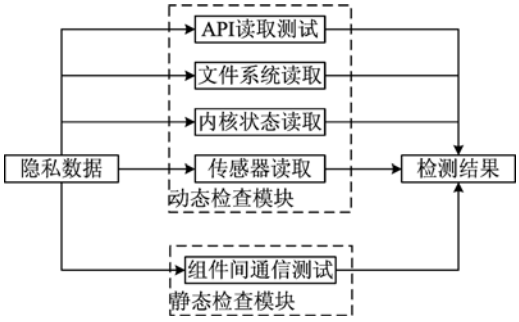

图 3 隐私读取检测

Fig. 3 Evaluation of privacy reading

首先，PrivacyMiner 检测所有可以读取隐私数据的 API 是否受到权限的保护，这一检测由动态检查模块完成. 该模块在没有声明相应权限的情况下，尝试调用所有读取隐私数据的 API，查看是否能够读取隐私. Android 系统中，应用软件在没有声明权限的情况下调用需要权限的 API，就会列出安全异常，这一异常可由动态检查模块捕获.

其次，PrivacyMiner 检测隐私数据能否通过内核状态或者文件系统读取. Android 系统中的大多数隐私数据，包括联系人、短信和通话记录，都以 sqlite 数据库的形式保存在特定的文件中，然后使用 Content Provider 组件提供数据访问的接口，并进行相应的权限约束. 虽然其他软件在访问 Content Provider 组件的数据时，Android 系统会进行权限检查，但由于这些数据保存在文件系统中的一个特定文件中，恶意软件会尝试读取这些文件，从而绕过 Content Provider 的权限约束. 此外，恶意软件还会通过某些特定的系统状态，来获取隐私数据，因此 PrivacyMiner 的动态检查模块，也会对其进行检测.

Android 设备中的某些传感器，会泄漏用户隐私，如 GPS 传感器或 NFC 设备. PrivacyMiner 的动态检查模块会列出包含隐私数据的设备或传感器，检查它们是否得到了有效的保护.

最后，PrivacyMiner 检查恶意软件是否能够通过组件间通信机制，调用原生软件的组件来读取隐私数据，这项检查通过静态污点分析模块完成. PrivacyMiner 先将所有的隐私数据标记为污点源，并将所有能将隐私数据传递给应用软件的接口，标记为汇聚点；然后检测是否存在从污点源到汇聚点的执行路径，并且执行路径上不包含任何权限检查，如 onTransact()、checkPermission()等函数.

4.2 隐私发送

当恶意软件成功读取用户的隐私数据后，它们就会希望把隐私数据发送出去. 我们只考虑三种隐私发送方式：电话、短信和网络发送. 我们忽略其他一些发送数据的方式，例如蓝牙、近场通信等，因为这些发送方式的传输距离有限，难以被利用.

如图 4 所示，PrivacyMiner 首先使用动态检查模块，来检测所有发送短信，拨打电话，或者发送网络数据的 API；然后再检测隐私数据能否通过组件间通信的方式发送出去，这项检查由静态污点分析模块完成：即先将所有应用软件能够传递给 Application Frame 或原生软件的数据标记为污点源，并将所有能够发送短信、拨打电话或者发送网络数据的接口标记为汇聚点. 通过使用污点分析模块，

检测是否存在从污点源到汇聚点的执行路径，并且执行路径上不包含任何权限检查.

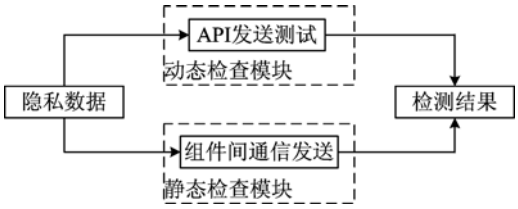

图 4　隐私发送检测

Fig. 4　Evaluation of privacy sending

5　实验验证

我们在 12 个版本的 Android 源代码上对 PrivacyMiner 工具进行实验验证，从 Android 2.1 到 Android 4.4.2.

5.1　隐私读取

在 22 个隐私类别中，PrivacyMiner 发现其中的 7 类没有受到有效的保护，如表 4 所示. 恶意软件可以在不声明任何权限的情况下读取这些隐私.

表 4　未受到有效保护的隐私类型

Tab. 4　Poorly protected privacy

隐私类型	相关保护机制
浏览器历史记录	READ_HISTORY_BOOKMARKS 权限
软件运行状态	GET_TASKS 等权限
Linux 内核状态	Linux 内核的访问控制机制
用户的位置信息	ACCESS_COARSE_LOCATION 等权限
隐私相关传感器	各种相关权限
已安装的应用软件	无
剪贴板	无

在这 7 类未受到有效保护的隐私类型中，前三类是由于 Android 的内核状态，特别是/proc 文件系统未受保护造成的. Android 系统要求应用软件必需声明 READ_HISTORY_BOOKMARKS 权限，才能读取浏览器的历史记录，但是由于恶意软件可以直接从/proc 文件系统中获取当前网络状态，包括正在连接的远程 IP 地址和端口，因此即使它们没有声明 READ_HISTORY_BOOKMARKS 权限，仍然可以获取到有关浏览器历史记录的隐私. 同样，恶意软件可以在没有声明 GET_TASKS 权限的情况下，从/proc 文件系统中获取正在运行进程的相关信息.

虽然 Android 系统要求应用软件必需声明 ACCESS_COARSE_LOCATION 或 ACCESS_FINE_LOCATION 权限，才能访问 GPS 模块提供的数据，但是其他一些与位置隐私相关的传感器并没有受到任何保护，如方向传感器，线性加速传感器和陀螺仪. 恶意软件可以通过这些传感器的数据，计算出用户移动的时间、距离、轨迹和方式.

此外，一些隐私未受到 Android 安全机制的任何保护，如已安装的应用软件和剪贴板. 恶意软件可以在用户不知情的情况下，获取用户喜欢哪些应用软件的信息以及这些应用软件是什么时候安装或卸载的. 如果用户在其他软件中复制电话号码、短信或密码，恶意软件也可以在用户不知情的情况下获取这些隐私数据.

5.2　隐私发送

在所有 12 个被测试的 Android 源代码中，PrivacyMiner 的污点分析模块都发现了一条通过网络发送隐私数据的执行路径. 该执行路径利用了原生浏览器中的一个组件，此组件可以接受来自恶意软件的指令，帮助它们把数据发送到外部服务器上，而不检查恶意软件是否具有相应的权限.

5.3　验证

在前面的实验中，我们发现了 7 类未受有效保护的隐私类型以及一条通过网络发送隐私数据的执行路径. 我们在 6 个 Android 设备上验证了这些漏洞，如表 5 所示，它们运行的系统版本从 Android2.1 到最新的 4.4.2. 结果表明，这些隐私问题在 6 个设备中都存在，并得到了 Android 安全团队的确认.

表 5　用于验证的 Android 设备

Tab. 5　Devices for verification

制造商	型号	Android 版本
Motolola	Droid	2.1
ZTE	Blade	2.3.7
ZTE	Grand X	4.0.4
Samsung	Galaxy S3	4.1.2
LG	Nexus 4	4.3
LG	Nexus 4	4.4.2

5.4　讨论

5.4.1　正确性

PrivacyMiner 是完备的，因为所有找出的系统漏洞，都是真实存在的，都可以被恶意软件利用，我们在 6 个 Android 设备上对这些漏洞进行了验证，如表 5 所示. 验证的结果表明，这些漏洞都真实存在于这些设备中. 另一方面，这些漏洞都得到了 Android 安全团队的确认，它们已经在 Android 系统中存在多

年,涉及当前几乎所有的 Android 设备.

PrivacyMiner 是非可靠的,PrivacyMiner 找不到漏洞的隐私读取或发送接口,并不一定不存在漏洞,因为 PrivacyMiner 中的某些步骤是由人工完成的,可能存在疏漏. 我们列出了 22 个类别的隐私数据,但这些类别可能没有涵盖所有的隐私,我们可能漏掉了某些类别的隐私数据. 此外,在动态检查模块的 API 读取测试中,我们先人工列出所有读取隐私数据的 API 函数,然后再测试它们是否受到相关的权限保护. 尽管我们人工列出了数百个 API 函数以及应用程序可以通过 Java 反射机制调用的隐藏 API 函数,但是,我们仍有可能漏掉某些 API 函数.

在 PrivacyMiner 给出的报告中,所有被找出的漏洞都是真实存在的,都能被恶意软件利用,那些报告受到妥善保护的隐私类别以及发送隐私的方式,仍可能存在某些未被覆盖的漏洞.

5.4.2 *改进方法*

为了更加有效地防止恶意软件读取并发送用户隐私,我们建议 Android 系统进行如下改进:

(Ⅰ) 不要为第三方软件开放某些内核状态,特别是/proc 文件系统.

(Ⅱ) 除了 GPS 模块之外,要为其他与用户隐私相关的传感器添加访问权限,包括方向传感器、线性加速传感器、陀螺仪、光线传感器等.

(Ⅲ) 由于剪贴板中经常包含用户的隐私数据,例如电话号码,短信内容等,我们建议 Android 系统为剪贴板添加一个新的访问权限.

(Ⅳ) 用户对应用软件的喜好以及他们什么时候安装或卸载了哪些应用,都应当受到 Android 安全机制的保护,我们建议 Android 系统添加一个新的权限来防止恶意软件读取这些隐私.

(Ⅴ) 为了防止恶意软件在读取隐私数据后,利用原生软件的组件将数据发送出去,我们建议浏览器在接收第三方应用软件的数据之前,先要检查这些软件是否声明了 Internet 权限,如果没有,应该先询问用户是否要继续访问网络.

6 结论

本文设计并实现了一个检测 Android 系统隐私保护机制的工具——PrivacyMiner,使用这一工具,我们将 Android 设备上的隐私数据划分为 22 个类型,并分别检测 Android 安全机制是否能有效地阻止它们被恶意软件所泄漏.

在实验中,我们对 12 个版本的 Android 源代码进行了检测,从 Android 2.1 到 4.4.2. 我们发现 7 个类别的隐私数据并没有受到有效保护,恶意软件可以读取这些隐私数据,并通过网络发送到外部服务器上,而无需声明相应的权限. 我们在 6 个 Android 设备上验证了这些隐私问题,并得到了 Android 安全团队的确认.

当前,PrivacyMiner 只针对 Application Framework 和原生软件进行检测,在以后的工作中,我们希望对第三方软件也进行检测. Android 市场中存在大量第三方软件,其中一些被广泛安装于用户的手机中,如果这些软件没有严格遵循 Android 的权限规则,就容易被恶意软件利用. 我们将把 PrivacyMiner 扩展为一个自动检测工具,用于检查第三方软件是否严格遵循了 Android 的安全规范,使得软件市场可以在开发者发布应用软件前,使用我们的工具进行安全检查.

参考文献(References)

[1] Apple Press Info. Apple' s App Store Marks Historic 50 Billionth Download [EB/OL]. http://www.apple.com/pr/library/2013/05/16Apples-App-Store-Marks-Historic-50-Billionth-Download.html.

[2] There have been 900 million Android activations, 48 billion app installs to date [EB/OL]. http://www.engadget.com/2013/05/15/900-million-android-activations/.

[3] Enck W, Gilbert P, Chun B, et al. TaintDroid: An information-flow tracking system for realtime privacy monitoring on SmartPhones [C]// Proceedings of the 9th USENIX conference on Operating systems design and implementation. Berkley, USA: ACM Press. 2010, 10: 255-270.

[4] Enck W, Octeau D, McDaniel P, et al. A study of android application security [C]// Proceedings of the 20th USENIX conference on Security. Berkley, USA: ACM Press 2011: 21.

[5] Orthacker C, Teufl P, Kraxberger S, et al. Android security permissions — Can we trust them? [C]// International ICST Conference on Security and Privacy in Mobile Information and Communication Systems. Aalborg, Denmark: Springer, 2012: 40-51.

[6] Zhou Y J, Jiang X X. Dissecting android malware: Characterization and evolution [C]// Proceedings of the 2012 IEEE Symposium on Security and Privacy. San Francisco, USA: IEEE Press, 2012: 95-109.

[7] Chia P H, Yamamoto Y, Asokan N. Is this app safe? A large scale study on application permissions and risk

signals [C]// Proceedings of the 21st International Conference on World Wide Web. Lyon, France: ACM Press, 2012: 311-320.

[8] Android Company. Permissions in Android [EB/OL]. http://developer. android. com/reference/android/Manifest. permission. html.

[9] Felt A P, Ha E, Egelman S, et al. Android permissions: User attention, comprehension, and behavior [C]// Proceedings of the 8th Symposium on Usable Privacy and Security. University of California, Berkeley, USA: ACM Press, 2012: Article No. 3(1-14).

[10] Lane M. Does the android permission system provide adequate information privacy protection for end-users of mobile apps? [C]// Proceedings of the 10th Australian Information Security Management Conference. Perth, Australia: ePrint, 2012: 66-73.

[11] Chin E. Felt A P, Greenwood K, et al. Analyzing Inter-application communication in android [C]// Proceedings of the 9th International Conference on Mobile systems, applications, and services. Bethesda, USA: ACM Press, 2011: 239-252.

[12] Kantola D, Chin E, He W D, et al. Reducing attack surfaces for intra-application communication in android [C]// Proceedings of the second ACM workshop on Security and privacy in SmartPhones and Mobile Devices. Raleigh, USA: ACM Press, 2012: 69-80.

[13] Grace M, Zhou Y J, Wang Z, et al. Systematic detection of capability leaks in stock android SmartPhones [C]// Proceedings of the 19th Network and Distributed System Security Symposium. San Diego, USA: ACM Press, 2012.

[14] Gibler C, Crussell J, Erickson J, et al. AndroidLeaks: Automatically detecting potential privacy leaks in android applications on a large scale [C]// Proceedings of the 5th International Conference on Trust and Trustworthy Computing. Vienna, Austria: Springer, 2012: 291-307.

[15] Wei X T, Gomez L, Neamtiu L, et al. Permission evolution in the android ecosystem [C]// Proceedings of the 28th Annual Computer Security Applications Conference. Orlando, USA: ACM Press, 2012: 31-40.

[16] Barrera D, Kayacik H G, van Oorschot P C, et al. A methodology for empirical analysis of permission-based security models and its application to android [C]// Proceedings of the 17th ACM Conference on Computer and Communications Security. Chicago, USA: ACM Press, 2010: 73-84.

[17] 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. Raleigh, USA: ACM Press, 2012: 217-228.

[18] Zhou W, Zhou Y J, Jiang X X, et al. Detecting repackaged SmartPhone applications in third-party android marketplaces [C]// Proceedings of the Second ACM Conference on Data and Application Security and Privacy. San Antonio, USA: ACM Press, 2012: 317-326.

[19] Zhou W, Zhang X W, Jiang X X. AppInk: Watermarking android apps for repackaging deterrence [C]// Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. Hangzhou, China: ACM Press, 2013: 1-12.

[20] Felt A P, Chin E, Hanna S, et ala. Android permissions demystified [C]// Proceedings of the 18th ACM Conference on Computer and Communications Security. Chicago, USA:ACM Press, 2011: 627-638.

[21] Chan P F, Hui C K, Yiu S M. DroidChecker: Analyzing android applications for capability leak [C]// Proceedings of the fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks. Tucson, USA: ACM Press, 2012: 125-136.

[22] Gibler C, Crussell J, Erickson J, et al. AndroidLeaks: Automatically detecting potential privacy leaks in android applications on a large scale [C]// Proceedings of the 5th International Conference on Trust and Trustworthy Computing. Vienna, Austria: Springer, 2012: 291-307.

[23] Enck W, Gilbert P, Chun B, et al. TaintDroid: An information-flow tracking system for realtime privacy monitoring on SmartPhones [C]// Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Vancouver, Canada: USENIX Association, 2010, 10: 255-270.

[24] Yang Z M, Yang M. Leakminer: Detect information leakage on android with static taint analysis [C]// Third World Congress on Software Engineering. Wuhan, China: IEEE Press, 2012: 101-104.

[25] Yan L K, Yin H. Droidscope: Seamlessly reconstructing the OS and dalvik semantic views for dynamic android malware analysis [C]// Proceedings of the 21st USENIX Security Symposium. Berkeley, USA: USENIX Association, 2012: 29.

[26] Yang Z M, Yang M, Zhang Y, et al. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection [C]// Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. Scottsdale, USA, ACM Press, 2013: 1 043-1 054.