Android 手机的轻量级访问控制 *

刘昌平[1], 范明钰[1], 王光卫[1], 郑秀林[2], 宫亚峰[3]
(1. 电子科技大学 计算机科学与工程学院, 成都 611731; 2. 北京电子科技学院, 北京 100070; 3. 国家信息技术安全研究中心, 北京 100091)

摘 要: Android 手机与计算机建立连接之后,计算机能以 root 身份登录 Android 手机,会给手机的应用程序和用户数据造成安全隐患。以 Android 手机为平台,提出了一种适用于 Android 手机的访问控制方法,该方法在 Android 手机的内核中增加访问控制模块,并根据手机用户定制的访问控制策略来控制计算机对 Android 手机的文件访问。仿真实验表明,该方法以较小的性能代价实现了 Android 手机的文件访问控制,能有效地保护手机用户的应用程序和数据安全。

关键词: 访问控制; 手机; 操作系统; 可信计算基
中图分类号: TP309 **文献标志码**: A **文章编号**: 1001-3695(2010)07-2611-03
doi:10.3969/j.issn.1001-3695.2010.07.059

Light-weight access control oriented toward Android

LIU Chang-ping[1], FAN Ming-yu[1], WANG Guang-wei[1], ZHENG Xiu-lin[2], GONG Ya-feng[3]
(1. *School of Computer Science & Engineering, University of Electronic Science & Technology of China, Chengdu* 611731, *China*; 2. *Beijing Electronic Science & Technology Institute, Beijing* 100070, *China*; 3. *National Research Center for Information Technology Security, Beijing* 100091, *China*)

Abstract: Computer user could login Android with root privilege after connection between computer and Android was built through USB interface. This privilege leakage exposed user's privacy date on cell phone to the potential attacker and trojan program. Aiming at this vulnerability, this paper proposed a light-weight method for controlling access on Android. This method embedded an access control module in the kernel of Android to control computer's access on Android according the access policies defined beforehand. Simulating experiment shows that this method can protect effectively user's privacy on cell phone with little performance cost.

Key words: access control; cell phone; operating system; trusted computing base

0 引言

随着网络与通信技术的不断发展,智能手机逐渐成为人们的日常消费品,智能手机的用户与日俱增。除了基本的通话功能以外,智能手机还具备 PDA(personal digital assistant)的主要功能,尤其包括个人信息管理以及无线接入互联网的功能,智能手机已经成为互联网中新型的终端节点。目前智能手机的主流操作系统主要有 Symbian、Windows Mobile 和 Android。

恶意程序威胁着计算机系统的安全,攻击者利用计算系统的漏洞在目标计算机上安装恶意程序,籍此来窃取被攻击者的敏感信息。智能手机通过在无线接入设备进入互联网的同时,也成为恶意程序新的攻击对象。智能手机的计算能力和存储空间有限,难以布署诸如防火墙、入侵检测系统等安全工具,智能手机中的用户隐私信息以及手机用户的合法权益成为攻击者新的攻击目标。

针对上述问题,本文以 Android 手机为平台,在手机的操作系统中增加轻量级访问控制模块,控制计算机对 Android 手机的文件访问。

1 Android 概述

1.1 Android

Android 是 Google 于 2007 年底发布的基于 Linux 内核的开源手机操作系统[1],2008 年 9 月 T-Mobile 正式发布了第一款 Google 智能手机 T-Mobile G1[2]。Android 的架构主要分为三部分:低层以 Linux 为操作系统内核,提供进程管理、存储管理等基本功能;中间层包括软件管理器、数据库服务等中间件,向应用层软件提供 API(application program interface);应用层以 Java 为开发语言,由第三方自行开发应用软件。2009 年 10 月 Google 发布了 Android SDK (software developing kits) 2.0[3],其中包括 Android 模拟器以及若干个开发工具,为第三方提供了软件开发空间。

1.2 ADB

Android debug bridge(ADB)[3]是 Android SDK 中的一个实用工具,向 Android 手机用户提供安装、管理应用软件和用户数据的功能。ADB 由客户端工具(ADB client, ADBC)和服务

收稿日期: 2009-11-19; **修回日期**: 2010-01-04 **基金项目**: 国家"863"计划资助项目(2009AA01Z403,2009AA01Z435)

作者简介:刘昌平(1975-),男,湖南祁阳人,博士研究生,主要研究方向为可信计算(gzucity@hotmail.com);范明钰(1962-),女,四川成都人,教授,博导,主要研究方向为信息安全;王光卫(1959-),男,四川成都人,高级工程师,博士,主要研究方向为信息安全、数字通信;郑秀林(1956-),男,教授,主要研究方向为密码算法设计与分析;宫亚峰(1951-),男,高级工程师,主要研究方向为信息安全、架构设计.

端工具(ADB daemon, ADBD)组成。客户端工具运行在计算机(如个人计算机)上,服务端工具以后台守护进程的方式运行在 Android 模拟器或手机上。ADBD 与 ADBC 建立连接之后,接收来自 ADBC 的命令,并在 Android 模拟器或 Android 手机上执行该命令。例如:

adb push newapp. apk /data/app/newapp. apk

该命令将计算机上的文件 newapp. apk 复制到 Android 模拟器或手机的目录/data/app 下,如果/data/app/newapp. apk 已经存在,ADBD 将覆盖原文件。

adb install newapp. apk

该命令在 Android 模拟器或 Android 手机上安装应用软件 newapp. apk,默认的安装路径是/data/app。

1.3 Android 安全

Android 为每个应用软件和该软件创建的数据文件分配一个 Linux ID,并为之创建一个沙箱(sandbox)来防止其他应用软件的干扰。在默认情况下,应用软件的活动范围仅限于自己的沙箱,没有权限执行对其他应用程序或操作系统有害的行为,如替换原来的程序文件等。例如 Android 仅允许应用软件/system/app/Contacts. apk 访问数据文件/data/data/com. android. providers. contacts/databases/contacts. db,并且拒绝其他应用软件对该数据文件的读写操作。

由此可见,Android 采用了进程隔离策略来保障应用软件及其数据文件的完整性和机密性。Android 的这种安全策略仅限于 Android 手机的内部范围,当 Android 手机通过 ADB 与计算机建立连接后,驻留在计算机上的恶意程序便会对 Android 应用软件和数据文件构成新的安全威胁。

2 Android 的安全隐患

Android 手机采用了进程隔离的安全策略来保障手机的应用程序和数据安全,即使手机用户无意中下载并执行了恶意程序,该恶意程序也没有足够的权限替换手机的程序文件。但是这种保护措施局限于 Android 手机内部,也就是说,Android 手机的可信计算基(trusted computing base,TCB)[4] 仅仅局限于手机本身。

手机用户通过 USB 接口将 Android 手机连接到计算机,此时手机的 TCB 应该扩展到所连接的计算机。然而 Android 手机所连接的计算机不一定是可信的,如图 1 所示。恶意程序不是运行在 Android 手机上,而是运行在计算机(如公共场所的计算机)上。攻击者首先在计算机上安装恶意程序,利用 ADB 探测 Android 手机并与之建立起连接,最后利用 ADB 向 Android 手机发布命令。

在 Google 提供的 Android SDK 2.0 中,ADB 是以 root 身份登录 Android 的操作系统,因此具有操纵 Android 手机文件的所有权限,给攻击者提供了植入木马程序和窃取手机用户隐私数据的机会。例如,攻击者在手机用户不知情的情况下,通过 ADB 向 Android 手机发布如下命令来替换原来的手机应用程序,将木马程序伪装成合法的应用程序:

adb push newapp. apk /data/app/newapp. apk

还可以发布如下命令来下载手机用户的数据库文件:

adb pull /data/data/com. android. providers. contacts/ databases/contacts. db contacts. db

Android 手机的数据库文件是以明文方式保存,攻击者下载了手机用户的数据文件之后便可以得到手机用户的隐私信息。

3 轻量级访问控制方法

3.1 方法的框架

本文的 Android 手机轻量级访问控制方法的框架由四部分构成,如图 2 所示。

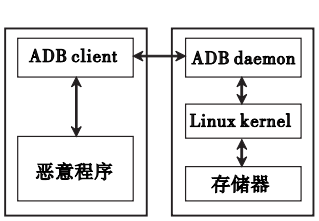

图1 Android的安全隐患

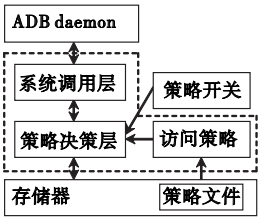

图2 Android访问控制框架

策略文件存储在 Android 手机的存储器中,该文件列出了 Android 手机的文件访问策略,由手机用户根据自己的需要加以定制,例如允许 ADB daemon 读程序文件,禁止写程序文件。

访问策略是位于 Linux 内核的存储空间,Android 手机启动时从存储器中读取策略文件来设置访问策略。

策略决策层位于 Linux 内核的系统调用层与存储器之间。当发生与文件或文件系统操作相关的系统调用时,策略决策层根据访问策略来控制 ADB daemon 对文件及文件系统的访问。

策略开关是一个在 Android 手机上新增的应用软件。当打开策略开关时,策略决策层根据访问策略来控制 Android 手机的文件访问;关闭开关时,策略决策层停止工作。手机用户在相对安全的场所(如家用计算机)关闭策略开关,备份 Android 手机的数据文件或者升级应用程序;接入公共场所的计算机时,打开策略开关,控制计算机对手机文件的访问。在缺省情况下,策略开关处于打开状态。

3.2 访问策略

访问策略描述了手机用户授予 ADB daemon 对 Android 手机文件的访问权限。手机用户对每个 Android 手机的文件建立一个访问策略,表示如下:

〈rights_list, filepath〉

其中:filepath 表示文件的路径及其名称;rights_list 表示 ADB daemon 访问文件的权限列表,是单个或多个权限的组合,表示为 rights_list = read | write | unlink | rename | create | null。

ADB daemon 对文件的访问权限表示如下:

read 为读文件权限;

write 为写文件权限;

unlink 为删除文件权限;

rename 为重命名文件权限;

create 为新建文件权限;

null 为禁止访问。

在 Android 手机中,目录/system/app 通常存放经 Google 认证的应用程序,目录/data/data 存放应用程序的数据库文件,目录/system/bin 和/sbin 存放少量 Linux 常用的命令行程序,目录/system/etc 存放 Linux 配置文件,目录/data/app 存放第三方应用程序。相对而言,这些文件比较固定,升级或更新的可能性较小,可以为这些文件设置如下的缺省文件访问策略:

〈read, /system/app/*〉
〈read, /system/bin/*〉
〈read, /sbin/*〉
〈read, /data/app/*〉
〈null, /data/data/*〉

Android 手机的目录/sdcard 通常存放手机用户的数据文件,目录/cache 存放临时文件。这些文件的更新比较大,可以设置如下这些文件的缺省访问策略:

〈read | write | rename | create, /sdcard/* 〉

〈read | write | unlink | create, /cache/* 〉

除了上述缺省的文件访问策略之外,手机用户通过图 2 中的策略开关定制文件访问策略。

3.3　访问控制

Linux 内核通过系统调用(system call)向用户态进程提供服务接口[5],Android 的中间层软件最终也是要通过 Linux 的系统调用来实现。在 Linux 内核中,与 3.2 节所列文件访问权限有关的系统调用[6]主要有以下五个接口:

sys_read( )为读文件的系统调用;

sys_write ( )为写文件的系统调用;

sys_unlink ( )为删除文件的系统调用;

sys_rename ( )为修改文件名的系统调用;

sys_create ( )为新建文件的系统调用;

sys_open ( )为打开文件的系统调用。

本文将上述系统调用称为 Android 手机文件访问系统调用,图 2 的策略决策层的访问控制流程如图 3 所示。

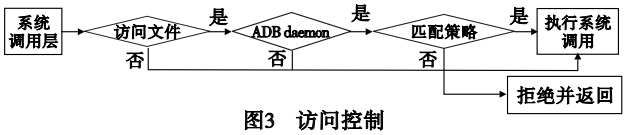

图3　访问控制

当发生系统调用时,首先检查当前系统调用是否属于 Android 手机文件访问系统调用。如果不是,则说明当前系统调用没有访问文件,执行流程按原来的顺序进行。如果发生了文件访问系统调用,则检查当前访问文件的主体是否是 ADB daemon。如果不是,则说明当前文件访问系统调用是由 Android 手机内部的应用程序产生的,仍然按照原来的顺序执行。如果 ADB daemon 产生了文件访问系统调用,则说明 ADB daemon 已经与计算机建立了连接。此时根据当前系统调用的参数(主要是访问方式和文件路径)与事先定义的访问策略进行匹配。如果匹配,则说明当前文件访问符合手机用户定制的访问策略,执行该文件访问操作;否则终止当前系统调用并返回。

4　仿真实验

4.1　实验环境

仿真实验以 Android SDK 2.0 提供的 Android 模拟器为手机平台,Android 的 Linux 内核版本是 2.6.27。根据第 3 章所述的访问控制方法,本文在 Android 的 Linux 内核中增加文件访问控制模块,并定制缺省的文件访问控制策略。为了与 Google 提供的 Linux 内核相区别,将 Android 原内核[7]简称为 OLK(original Linux kernel),将增加了访问控制方法的内核称为 ELK(extended Linux kernel)。

仿真实验由两部分组成,即功能测试和性能测试。功能测试的目标是在以 ELK 为内核的 Android 手机中 ADB 对手机文件的访问是否满足手机用户定制的访问策略;性能测试的目标是测试 ELK 的性能消耗。

4.2　功能测试

以 ELK 为内核启动 Android 模拟器,通过 ADB 向 Android 模拟器发布如下命令:

命令 1　adb push myphone. apk /system/app/Phone. apk

命令 2　adb pull /data/data/com. android. providers. contac ts/databases/contacts. db contacts. db

命令 1　试图以应用程序 myphone. apk 伪装成原来的应用程序 Phone. apk,命令 2 试图下载手机用户的数据库文件 contacts. db。上述命令对 Android 手机的文件访问都违反了缺省的访问策略。执行结果如图 4 所示。

```
E:\android2\tools>adb push myphone.apk /system/app/Phone.apk
failed to copy 'myphone.apk' to '/system/app/Phone.apk': Operation not permitted

E:\android2\tools>adb pull /data/data/com.android.providers.contacts/databases/c
ontacts.db contacts.db
failed to copy '/data/data/com.android.providers.contacts/databases/contacts.db'
: Operation not permitted
```

图4　功能测试的结果

4.3　性能测试

4.3.1　内存开销

缺省的策略文件定义了 Android 模拟器中大部分文件的访问策略,共计 4 535 个访问策略。在 Android 模拟器启动时,ELK 将该访问策略加载到 Linux 的内核空间。本文对比了 OLK 和 ELK 的内存消耗,如表 1 所示。

表 1　内存消耗　　KB

OLK		ELK	
占用的内存	空闲内存	占用的内存	空闲内存
83 384	10 760	84 056	10 088

从表 1 可以看出,增加访问控制模块之后,Linux 内核的内存消耗增加了 672 KB,相对于 OLK 的内存消耗而言,仅增加了 0.81%,这说明本文的访问控制方法对 Android 模拟器的内存额外开销很小。

4.3.2　CPU 开销

采用以下两种方案测试本文的 Android 手机访问控制方法的 CPU 开销。

方案 1　计算机通过 ADB 每隔 1 s 向 Android 模拟器发布命令:adb push myphone. apk /system/app/Phone. apk,测试 OLK 和 ELK 的 CPU 占用率。

方案 2　计算机通过 ADB 每隔 10 s 向 Android 模拟器发布命令:adb push myphone. apk /system/app/Phone. apk,测试 OLK 和 ELK 的 CPU 占用率。

统计两种实验方案在 10 min 的时间范围内 Linux 内核对 CPU 的占用率,如图 5 和 6 所示。从总体上来看,ELK 和 OLK 的 CPU 占用率没有显著的差别。在方案 2 的实验数据中,ELK 和 OLK 的最大 CPU 占用率相差 1.36%,其他的 CPU 额外开销不超过 0.5%,这说明本文的访问控制方法不会给 Android 模拟器造成明显的性能消耗。

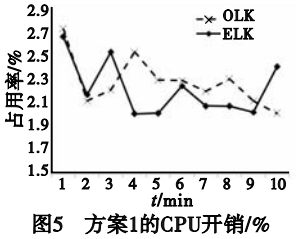

图5　方案1的CPU开销/%

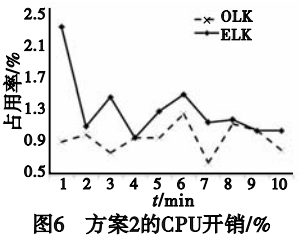

图6　方案2的CPU开销/%

5　结束语

智能手机的计算能力和存储空间有限,常用的安全保障措施难以适用于智能手机,手机用户的隐私信息 (下转第 2628 页)

$0.9898\times0.9877\times0.9860\times0.9881\times0.9898\times0.9870\approx0.9305$

下面分析算法的计算复杂性。

步骤 b)对 $x_{0,16}$ 的所有可能值依次进行穷举,故步骤 b)的计算复杂性约为 2^{16},而 b)中保留下来的 $x_{0,16}$ 的候选值的个数约为 $2^{16-7}=2^9$。步骤 c)以 2^9 个 $x_{0,16}$ 为基础,穷举 $x_{0,24}$ 的所有可能值,故步骤 c)的计算复杂性约为 $2^9\times2^8=2^{17}$,而 c)中保留下来的($x_{0,16}$,$x_{0,24}$)的候选值的个数约为 $2^{24-15}=2^9$。步骤 d)以 2^9 个($x_{0,16}$,$x_{0,24}$)为基础,穷举 $x_{0,32}$ 的所有可能值,故步骤 d)的计算复杂性约为 $2^9\times2^8=2^{17}$,而 d)中保留下来的($x_{0,16}$,$x_{0,24}$,$x_{0,32}$)的候选值的个数约为 $2^{32-23}=2^9$。步骤 e)以 2^9 个($x_{0,16}$,$x_{0,24}$,$x_{0,32}$)为基础,穷举 $x_{0,40}$ 的所有可能值,故步骤 e)的计算复杂性约为 $2^9\times2^8=2^{17}$,而 e)中保留下来的($x_{0,16}$,$x_{0,24}$,$x_{0,32}$,$x_{0,40}$)的候选值的个数约为 $2^{40-31}=2^9$。步骤 f)以 2^9 个($x_{0,16}$,$x_{0,24}$,$x_{0,32}$,$x_{0,40}$)为基础,穷举 $x_{0,48}$ 的所有可能值,故步骤 f)的计算复杂性约为 $2^9\times2^8=2^{17}$,而 f)中保留下来的($x_{0,16}$,$x_{0,24}$,$x_{0,32}$,$x_{0,40}$,$x_{0,48}$)的候选值的个数约为 $2^{48-39}=2^9$。步骤 g)以 2^9 个($x_{0,16}$,$x_{0,24}$,$x_{0,32}$,$x_{0,40}$,$x_{0,48}$)为基础,穷举 $x_{0,56}$ 的所有可能值,故步骤 g)的计算复杂性约为 $2^9\times2^8=2^{17}$,而 g)中保留下来的($x_{0,16}$,$x_{0,24}$,$x_{0,32}$,$x_{0,40}$,$x_{0,48}$,$x_{0,56}$)的候选值的个数约为 $2^{56-47}=2^9$。步骤 h)以 2^9 个($x_{0,16}$,$x_{0,24}$,$x_{0,32}$,$x_{0,40}$,$x_{0,48}$,$x_{0,56}$)为基础,穷举 $x_{0,64}$ 的所有可能值,故步骤 h)的计算复杂性至多为 $2^9\times2^8=2^{17}$。综上所述,对迭代模型的分割攻击算法的计算复杂性约为 $2^{16}+2^{17}+2^{17}+2^{17}+2^{17}+2^{17}+2^{17}\approx2^{19.7}$。

由上面的分析知,算法的存储复杂性约为 $2^9+2^9+2^9+2^9+2^9+2^9\approx2^{11.6}$。

由于攻击算法仅利用了 1 个明密对所对应的量化序列,故数据复杂性仅为 1 个明密对。证毕。

定理 4 在密钥长度为 64 bit 时,分割攻击可以0.930 5 的成功率将穷举攻击 x_0 的密钥熵由 64 bit 降至 19.7 bit,降低了 44 bit,并且算法的计算复杂性和存储复杂性都可在 PC 机上实现。

对于 XW 算法来说,利用上述分割攻击算法就可在选择明文攻击条件下由量化序列 $\{b_i\}_{i=1}^{128}$ 求得混沌状态 x_N 的值。利用 $f(x)=4x(1-x)$,可由 x_N 的值求得 x_{N-1}(当 $x_{N-1}\neq1$ 时)的两个可能值,再由 x_{N-1} 的每个可能值进一步求出 x_{N-2} 的所有可能值,依此类推。由于参数 N 已知,只要逆推 N 层就可求出初态 x_0 的所有可能取值集合。另一方面,由于混沌状态 x_N 之前的状态均已丢弃,从信息论的角度来说,不论用何种方法均无法惟一确定 x_0 的值。

4 结束语

本文分析了一类混沌迭代分组密码算法的安全性,发现了该加密算法的信息泄露规律,给出了选择明文攻击条件下的分割攻击方法,说明该加密算法是不安全的。该算法不能抵抗分割攻击的根本原因在于:当初始值的若干低位修改为 0 时,新初始值产生的密钥流序列与原初始值产生的密钥流序列具有明显的不独立性。因此在设计混沌密码算法时,应确保在初始值和参数的低位发生变化时,新产生的密钥流序列与原密钥流序列近似独立。

参考文献:

[1] 颜森林. 光纤混沌双向保密通信系统研究[J]. 电子学报,2005,33(2):266-270.

[2] 徐淑奖,王继志. 一类改进的混沌迭代加密算法[J]. 物理学报,2008,57(1):37-41.

[3] 胡汉平,刘双红,王祖喜,等. 一种混沌密钥流产生方法[J]. 计算机学报,2004,27(3):408-412.

[4] ALVAREZ E, FERNANDEZ A, GARCIA P, *et al*. New approach to chaotic encryption[J]. Physics Letters A,1999,263:373-375.

[5] 李树钧,牟轩沁,纪震,等. 一类混沌流密码的分析[J]. 电子与信息学报,2003,25(4):473-479.

[6] 金晨辉,高海英. 对两个基于混沌的序列密码算法的分析[J]. 电子学报,2004,32(7):1066-1070.

[7] WANG Xin-gang, ZHAN Meng, LAI C H, *et al*. Error function attack of chaos synchronization based encryption schemes[EB/OL]. http://arxiv.org/nlin.CD/0305015.

[8] SHORT K M. Signal extraction from chaotic communications [J]. International Journal of Bifurcation and Chaos,1997,7(7):1579-1597.

[9] LI Shu-jun, CHEN G R,ALVAREZ G. Return-map cryptanalysis revisited[EB/OL]. http://arxiv.org/nlin.CD/0501018.

(上接第 2613 页)容易受到攻击者的窃取和破坏。本文以 Android 智能手机为平台,提出了一种轻量级的手机文件访问控制方法,该方法解决了 Android 手机与计算机建立连接时存在的安全隐患。手机用户利用该方法定制手机文件访问策略,能够较好地保护手机应用程序的完整性和隐私信息的机密性。

此外,基于 Windows Mobile 的智能手机采用 Microsoft Active Sync 同步软件,基于 Symbian 的智能手机通常采用 PC Suite 同步软件,这些同步软件与 Android 的 ADB 存在类似的安全问题,将本文方法应用于其他类型的智能手机是下一步的研究工作。

参考文献:

[1] Google. Android official website[EB/OL]. [2009-11-10]. http://www.android.com.

[2] T-Mobile. The T-Mobile G1 with Google phone official site[EB/OL]. [2009-11-10]. http://www.t-mobileg1.com/.

[3] Google. Android software developing kits 2.0[EB/OL]. [2009-11-10]. http://developer.android.com/sdk/.

[4] Trusted Computing Group. TCG architecture overview[EB/OL]. [2009-11-10]. https://www.trustedcomputinggroup.org/groups/TCG_1_4_Architecture_Overview.pdf.

[5] STONES R, MATTHEW N. Beginning Linux programming[M]. 2nd ed. Birmingham: Wrox Press Ltd, 2003.

[6] BOVET D P, CESATI M. Understanding the Linux kernel[M]. 3rd ed. Sebastopol,CA: O'Reilly Media, Inc,2005.

[7] Linux Kernel Organization Inc. Git[EB/OL]. [2009-11-10]. http://android.git.kernel.org/.