Android 安全加固技术①

乜聚虎[1,2]，周学海[1,2]，余艳玮[1,2]，吴志忠[1,2]

[1](中国科学技术大学 计算机科学与技术学院，合肥 230027)
[2](中国科学技术大学 苏州研究院，苏州 215123)

摘　要：为了保护 Android 智能手机安全，在深入分析 Android 系统安全机制的基础上，提出了一个基于强制访问控制的安全加固技术。该技术首先通过修改 Android 内核，添加一个平台无关的强制访问框架，其次为进程、文件等系统对象添加安全属性，最后通过制定一套细粒度的强制访问规则，对应用程序实施强制访问控制。通过在模拟环境中的测试，验证了该技术可以有效保护 Android 系统安全。

关键词：安全加固；强制访问控制；Android

Android Security Reinforcement Technology

NIE Ju-Hu[1,2], ZHOU Xue-Hai[1,2], YU Yan-Wei[1,2], WU Zhi-Zhong[1,2]

[1](Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
[2](Suzou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215123, China)

Abstract: To protect the Android-powered smartphones, a security reinforcement technology based on mandatory access control is proposed. It is implemented as follows: first, a platform-independent mandatory access control framework is added to the Android kernel; then, security attributes are attached to system objects; finally, a set of fine-grained access rules are made to control applications' permissions. The effectiveness of the technology is proved through the test in an emulator environment.

Key words: security enforcement; mandatory access control; Android

1　引言

近年来，智能手机在全球范围内迅速普及。市场研究公司 Gartner 发布的数据显示，2010 年前三季度，全球智能手机销量同比增长超过 50%，智能手机正在占据越来越多的手机市场份额。

智能手机不仅是通话设备，同时也是一种计算设备，通过在手机操作系统上安装第三方软件，其功能非常丰富，很多人开始使用智能手机处理日常事务，如收发邮件，电子支付等。这样智能手机不可避免的需要访问、存储用户的各种账户、密码等关键数据，因而极易成为恶意软件攻击的对象。因而与非智能手机相比，智能手机系统存在更多的安全威胁。从 2004 年第一个手机病毒出现起[1]，现在已发展有数百种，而且还在继续增长，智能手机安全将面临恶意软件的严重威胁。

Android 最早由 Google 公司于 2007 年发布，目前由开放手机联盟（OHA）开发维护，是一个完整、开放、免费的手机平台[2,3]。与其他智能手机系统相比，Android 为应用开发者提供了更多的功能接口，其中包括很多系统底层接口，提高了系统的可扩展性，但同时也为恶意软件提供了便利，针对 Android 系统的木马等恶意软件也更容易被实现。根据网秦发布的智能手机病毒数据[4]（表 1 所示），目前已有的针对 Android 系统的恶意软件大多采用伪装的方式，骗取用户安装并授予一定的权限，之后滥用这些权限在后台执行一些特定行为，包括窃取用户隐私或通过发送短信等方式消耗用户资费等，给用户造成损失。

本文在深入分析了 Android 系统的安全机制基础上，提出了一种安全加固技术，通过对应用程序权限的细粒度的强制访问控制，限制恶意软件滥用权限。

① 收稿时间:2011-02-19;收到修改稿时间:2011-03-21

在模拟环境中对系统进行了测试， 结果表明，该技术可以有效防止恶意软件的危害，保护系统的安全。

表 1 Android 恶意软件

类别	危害方式	典型恶意软件
流氓软件类	难以卸载，或影响手机正常使用等	MSO.PJApps.ZA
消耗资费类	通过后台联网、订购业务等恶意消耗用户资费	SW.Mobispy.A
隐私窃取类	窃取用户私人信息或文件以及用户设备信息等	BIT.GeiNiMi.X
后门软件类	隐藏在后台提供远程访问或操作手机的接口等	BIT.GeiNiMi.V

2 Android安全机制

Android 系统基于 Linux 内核，采用了分层的体系结构，自底向上包含五层，分别是 Linux 内核、本地库、Android 运行时环境、应用框架和应用程序。其中底层的 Linux 内核和上层的应用框架层分别包含了一种安全机制，在不同层次上控制应用程序的访问权限。如图 1 所示。

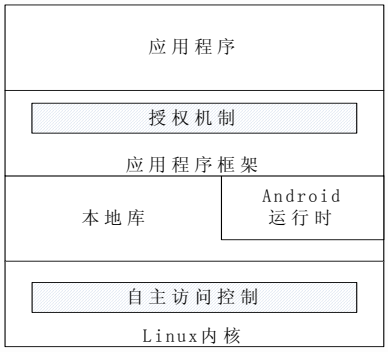

图 1 Android 两层安全机制

2.1 Linux 内核层安全机制

Linux 内核为 Android 系统提供了在桌面电脑和服务器系统中广泛使用的自主访问控制机制。系统中每个主体（进程等）都属于一个用户，并属于一个或多个用户组，每个客体（文件等）都有一个限定主体的用户或组对其访问权限的访问控制列表。对于每一次访问操作，系统检查客体的访问控制列表，判断主体是否具有对客体的访问权限。

自主访问控制存在两个主要的缺陷。首先，系统中存在一个超级用户，具有超级用户权限的主体不受任何自主访问控制机制的约束，可以对客体执行任何该客体所支持的操作。包括智能手机系统在内的现代计算机系统越来越复杂，其中不可避免的会存在一些漏洞，一些恶意软件可以利用这些漏洞，获取系统的超级用户权限，进而肆意窃取或破坏系统数据。其次，自主访问控制机制的控制粒度较大，客体的访问控制列表只区分三种主体：属于同一用户的主体、属于同一用户组的主体以及其他主体。这样如果一个客体允许其他主体对自身的某一个访问权限，就意味着系统中的所有主体都具有对该客体的这一访问权限，这就给一些恶意程序提供了可乘之机，访问他们本不该访问的权限。

2.2 应用框架层安全机制

Android 在应用框架层提供了一种授权机制。应用框架定义了若干对系统的访问权限，如访问网络的权限，接收或发送短信的权限等。默认情况下，应用程序不能访问这些权限，需要在安装的时候向系统请求需要的权限，系统请用户选择是否授予相应的权限。

应用框架层的授权机制同样存在粒度较大的问题。例如，如果一个程序获得了 INTERNET 权限，则该程序可以访问系统中网络相关的所有权限，可以访问任何端口，使用各种协议等，这也为系统安全带来了隐患，很多恶意都利用了这一漏洞。

3 基于强制访问控制的安全加固技术

Android 安全机制所存在的问题主要是超级用户权限的存在和访问控制粒度较大。这两个问题都可以通过添加强制访问控制机制来适当解决。平台无关的访问控制框架(Platform-independent and Flexible Access Control Framework,简称 PIFAC)是一个可移植的强制访问控制框架[5]，本文研究了在 Android 系统中应用 PIFAC，以细化访问控制粒度，增强系统安全的技术。

3.1 强制访问控制

强制访问控制是一种应用于自主访问控制之上，进一步限制访问权限，增强系统安全性的安全技术，已在桌面和服务器等系统上得到了广泛使用[6]。

在强制访问控制理论中，系统对象可分为客体和主体两种。客体是指系统中一切可保存信息的实体和其他资源等，如目录、文件以及网络端口等；主体指能够对客体发起访问的实体，主要是进程等。每个客体和主体拥有一个安全属性。一套可配置的策略规定了主体如何对客体进行操作，当某个主体对一个客体发起一项访问时，系统分别获取主体和客体的安全属性，并查询策略，判断是否允许该操作的执行。

由于该机制可以精确控制每个进程对文件等客体的访问权限，因而大大细化了系统访问控制的粒度。同时由于安全策略的可配置性，使得对系统安全的配置更加灵活，便于对系统的维护和管理。

3.2 PIFAC

PIFAC 是一个跨平台的基于强制访问控制的安全模块，已被应用于 Windows、Linux 等桌面和服务器系统以及 Rterms 等嵌入式系统上，它对平台特征进行了抽象，因而可以方便的移植到新的平台。

PIFAC 由三个部分组成：安全服务器、操作系统 Hook 层和策略。其中安全服务器主要实现策略的存储和判断，是一个平台无关的部分；操作系统 Hook 层的作用是截获系统中主体对客体的访问，不同系统上有不同的实现；策略规定了系统中主体对客体的访问权限，在 Android 系统上需要实现一个 Hook 层，并根据系统配置制定相应的策略。

Android 系统包含两类应用程序：底层应用程序，如 ls 等命令行程序，一般由 C/C++实现，不与用户直接交互；上层应用程序，如通讯录等直接与用户交互的程序，由 Java 语言实现，运行于 Delvik 虚拟机中。PIFAC 对于底层应用程序的处理与 Linux 系统中类似，而上层应用程序由于是在 Delvik 虚拟机中执行，所以需要进行特殊处理，具体的细节将在下一节说明，本节给出 PIFAC 在 Android 上实现的框架，如图 2 所示。

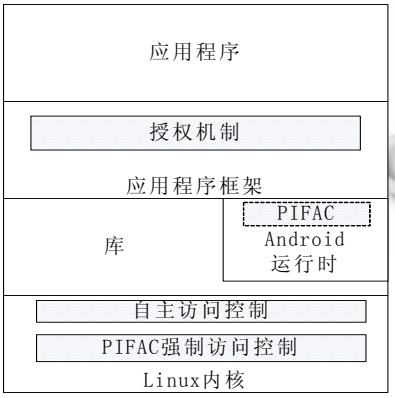

图 2　PIFAC 实施框架

4　实现

4.1 在 Android 系统上实现 PIFAC

4.1.1 添加内核对 PIFAC 的支持

PIFAC 以内核补丁的形式提供，并不是 Android 所运行的内核的一部分，另外 Android 所运行内核不支持安全模块。所以首先将 PIFAC 代码加入到内核代码的相应目录，然后重新配置内核以打开对安全模块的支持选项，如图 3 所示，最后重新编译得到支持 PIFAC 安全模块的内核。

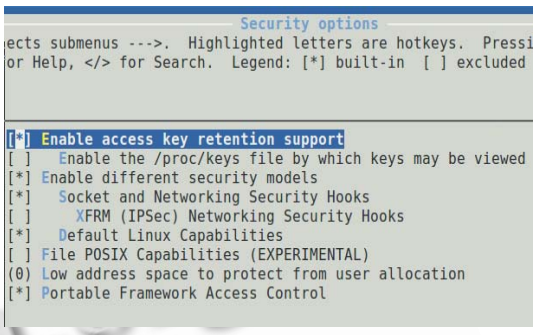

图 3　内核配置选项

4.1.2 装载策略

策略是实施强制访问控制的依据，需要在系统启动的时候加载入内核。为此，在系统根目录下添加策略文件，并修改 Android 的启动过程，使其在启动的时候加载策略文件。策略文件的加载过程如图 4 所示。

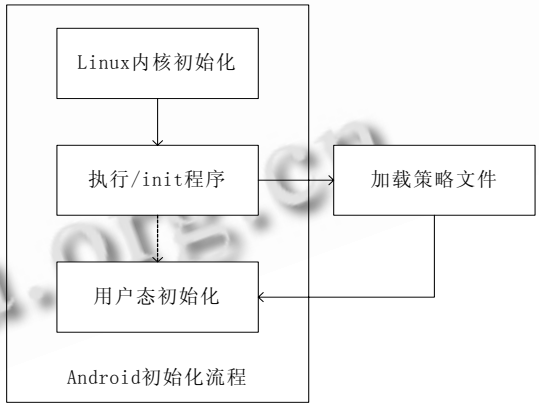

图 4　策略加载过程

4.1.3 给进程和文件打标签

访问控制中的主体是进程。在 Linux 内核中，只有 init 进程是显式的赋予其安全标签的，而其他的所有进程都是通过继承父进程的安全标签或通过标签的转移规则等方式自动获取安全标签的。

访问控制的客体是文件。我们使用内存系统来存放文件的安全标签：在系统根目录下创建文件标签文件/pfac/file_contexts，如图 5，然后修改/init 文件，使其加载完策略之后马上加载文件标签文件。

```
/data/org.pfac.create_file dte_context(create_dir_dir_t)
/data/org.pfac.create_dir dte_context(create_dir_dir_t)
/data/org.pfac.delete_file/delete_file_test
dte_context(delete_dir_dir_t)
/data/org.pfac.delete_dir/delete_dir_test
dte_context(delete_dir_dir_t)
/data/org.pfac.read_file dte_context(read_file_file_t)
/data/org.pfac.write_file dte_context(write_file_file_t)
/data/org.pfac.delete_dir/delete_dir_test
dte_context(delete_dir_dir_t)
```

图 5 PIFAC 标签文件

4.1.4 上层应用程序进程安全域的转移

Linux 系统中进程安全域的转移是在执行程序的系统调用服务例程 do_exec()函数中完成的。而 Andorid 上层应用程序的进程在 Delvik 虚拟机中执行，不会经过 do_exec()服务例程，因而需要其他方式完成进程域的转移。受 init 进程启发，我们通过显式的为进程赋予安全标签来实现进程域的转移：在 Delvik 虚拟机读取一个新的程序并准备执行时，我们查询策略数据库，获取新进程应该转移成的目标域的安全标签，然后显式的赋予该 Delvik 虚拟机目标域的安全标签。

4.2 测试

使用 FakeSpy.A 和 Consumer.X 两个恶意软件进行测试。FakeSpy.A 是一个间谍软件，它伪装成一款通讯录软件，诱骗用户安装。一旦完成安装，该软件便定期的向远程服务器 2222 端口发送用户通讯录中的数据，泄露用户的隐私。Consumer.X 是一个消耗资费类软件，它伪装成一个计算器程序，每当用户点击按键 2，该程序即开始通过 6583 端口访问远程服务器，并产生较大的网络流量，从而消耗用户资费。

根据两个软件的说明可知，它们不需要访问网络，不会产生数据资费。所以分别定义两条策略：禁止 FakeSpy.A 访问所有端口为 2222 的网络地址（也可以设置为禁止访问所有端口的地址，这里为了比较，仅禁止 2222 端口），以及禁止 Consumer.X 访问 6358 端口地址。分别在未使用和使用了 PIFAC 的系统上，以及添加和未添加策略的系统上对软件进行测试，结果显示，在未使用 PIFAC 的系统上，或使用了 PIFAC 但没有添加策略的系统上，两款恶意软件都会对系统造成危害，而在使用了 PIFAC 并添加了相应策略的系统上，两款恶意软件都无法再通过网络连接对系统造成危害，测试结果如表 2 所示。

表 2 测试结果

	未使用 PIFAC	使用 PIFAC	
		无策略	有策略
FakeSpy.A	造成危害	造成危害	无法危害
Consumer.X	造成危害	造成危害	无法危害

测试结果表明，使用了 PIFAC 并正确配置策略的系统可以有效防止恶意软件所造成的危害。

5 结语

Android 是一个备受关注的智能手机系统，本文深入分析了它的安全机制，针对其自身安全机制所存在的一些问题，提出了一个使用 PIFAC 强制访问控制框架增强系统安全的方法。该方法首先修改 Android 系统所使用的 Linux 内核，添加内核对 PIFAC 模块的支持，然后为系统中的主体和客体添加安全属性，最后为系统制定安全策略用以描述细化的访问控制规则。通过在模拟器中对恶意软件的测试，验证了该技术的有效性。下一步的工作重点是研究安全策略的制定和自动配置方法，由于安全策略的粒度决定了强制访问控制规则，并且需要根据系统配置的变化而改变，因而如何自动根据系统配置制定有效的安全策略，也是一个要解决的重要问题。

参考文献

1 Schmidt AD, Schmidt HG, Batyuk L, Clausen JH, Camtepe SA, Albayrak S, Yildizli C. Smartphone malware evolution revisited: Android next target? Proc. of the 4th IEEE International Conference on Malicious and Unwanted Software. 2009. 1–7.

2 Android Open Source. http://source.android.com

3 Android Developers. http://androidappdocs. appspot. com/index.html

4 网秦安全播报. http://virus.netqin.com/android/

5 胡大磊,周学海.平台无关的访问控制框架研究与实现.计算机系统应用,2010,19(3):17–20.

6 刘克龙,冯登国,石文昌.安全操作系统原理与技术.北京:科学出版社,2004.90–96.