**针对Android文件管理器删除数据的安全性研究**

摘 要 随着移动互联网的飞速发展，智能移动终端已经成为人们个人隐私数据的重要载体之一。但是对于目前的移动智能终端来说，系统所提供的数据删除接口并没有达到预想中的安全性。在现有的Android设计模型特别是安全模型中存在一些难以回避的缺陷，正是这些缺陷导致了这些不易察觉的残留数据存在风险，而这些风险被以往的安全分析所忽略，甚至主观的认为这些风险不存在。在这种情况下，攻击者能够利用安全防范的疏忽发起攻击，进而更为严重地威胁到设备相关使用者的隐私和数据安全。以Android系统为目标，利用AOP切片技术，对存在于Android系统中的多种文件管理器删除数据的方法进行了研究，研究表明其中部分方法存在着严重地安全隐患。着重分析了Android用户比较常用的通过文件管理器删除数据的操作。实验成功地从Android设备中恢复出大量的通过上述方式删除的数据，证明了目前的文件管理器对数据的删除是不安全的。提出名为PureEnc的解决方案来保护Android设备中的数据以对抗数据残留问题。PureEnc可以实现对Android应用程序的自动修改，以达到保护数据的目的，不需要对Android系统进行修改，可以很方便地部署并有效地保护应用程序的数据，以对抗数据残留漏洞。

关键词 Android，数据删除，数据残留

1 引言

移动只能终端设备的流行改变了人们处理个人信息的方式。越来越多的人选择使用手机等移动设备处理个人事务和工作事务。移动应用开发者们提供了各种各样的程序来帮助人们更好地处理生活中使用到的各类信息，比如使用社交应用来和朋友保持联系，使用银行应用来处理自己的银行事务，使用照相机应用来记录生活中的美好瞬间。这种趋势的出现导致了移动智能终端设备中存储了越来越多的和用户紧密相关的个人隐私数据，同时也导致了个人隐私数据泄露的事件频频发生。

作为目前最流行的移动智能终端操作系统，在很长一段时间内，Android都是个人隐私数据泄露事件发生的主要源头[1]，导致这种情况出现的主要原因在于Android系统和应用程序对于个人隐私数据的处理存在安全隐患，尤其是在进行数据删除操作时，没有满足预先设想的安全性。虽然在闪存介质[2]上的数据残留问题已经被研究得很多了，但是在Android系统中，这个问题仍然没有得到文件系统和操作系统的重视。在Android应用程序的开发文档中，并没有针对该问题进行特殊的说明，也没有警告开发者使用一般数据删除接口时可能会存在的安全问题，这导致了绝大部分Android应用程序都使用了不安全的数据删除接口处理敏感的数据，最终导致了数据泄露的发生。

本文旨在对Android系统上使用文件管理器进行数据删除的接口的安全情况进行深入的调查研究，并针对可能出现的安全问题制定合理的解决方案。

2 背景

2.1 Android数据存储

和传统的桌面电脑不同，搭载Android系统的智能移动终端往往选择闪存作为自身的存储介质。对于闪存来说，最重要的一大特性是闪存上所有的数据读写都是以块为单位的。一般情况下，一个数据块的大小是几百到几千个字节。在进行数据删除操作时，即使要删除的数据只是一个块的一部分，也需要将整个块上的全部数据都擦除。因此，为了提高磁盘读写的效率，当删除数据时，闪存的控制芯片不会真正把数据块里的数据全部置0，而是将这个数据块标记为未使用，当下一次需要向该数据块内写入数据时，才真正把数据擦除，这就导致了在闪存介质上，被删除的数据会继续残留在介质上一段时间，这个时间和介质的大小以及磁盘读写的频率有关[3]。

2.2 数据擦除

一般来说，Android系统中有两种主要的数据擦除方式。第一种是通过调用系统API直接删除文件，通过这种方式删除的文件，其内容会继续残留在磁盘上。通过对磁盘进行全盘镜像提取，就可以获取这些残留的数据内容，之后通过一些取证手段就能够恢复出原有数据的内容。在Android系统中，绝大多数的数据擦除操作都属于这种方式，虽然在系统中呈现的接口不尽相同，但是在底层均是调用Linux删除文件的系统来实现文件的擦除。Android系统中另一种数据擦除方式就是恢复出厂设置，通过触发系统设置界面中的相关接口，Android系统会重启到Recovery模式。在Recovery模式下，系统会自动对整个userdata分区进行格式化。一般说来，恢复出厂设置的过程中，系统会调用闪存介质底层接口来重写整个分区，确保分区上所有的数据都被安全地擦除。

2.3 Android数据保护

Google从Android 3．0开始在Android系统中加入了可供选择的全磁盘加密功能，全磁盘加密功能基于Linux dm-crypt特性修改而来。用户可以在系统设置中选择为自己的设备开启全磁盘加密，开启了全磁盘加密后，每次系统启动前都需要用户输入用于解密的根密钥。全磁盘加密是一次性的不可逆的过程，这意味着一旦用户开启了全磁盘加密，除非格式化整个设备，否则没有其他的方法来解除磁盘的加密状态。

在Android系统中，为了减少全磁盘加密对系统性能的影响，Google去除了dm-erypt方案中所有用于对抗取证的安全策略，这在一定程度上削弱了全磁盘加密的安全性。另一方面，dm-crypt方案的on-the-fly eneryption/decryption特性决定了全磁盘加密无法对抗来自系统内部的攻击，当系统已经处于运行状态时，磁盘上所有的数据将不再收到全磁盘加密的保护，拥有数据访问权限的任何用户和App都能够读取解密过的数据。在Android设备root非常普遍的今天，全磁盘加密并不能有效地保护磁盘数据的安全。

此外，由于方案设计上的一些缺陷，导致了全磁盘加密并没有带来预想中的安全性。Android系统中的全磁盘加密密钥由用户的锁屏PIN码生成，同时加解密磁盘所需要的加密参数(encryption parameters)都存储在本地磁盘对应分区的尾部的crypto footer结构中，通过解析这个结构体，攻击者就能够获得对全磁盘加密保护的磁盘发起暴力穷举攻击所需要的参数，即穷举PIN码。同时，作为一个日常使用的手持设备，绝大多数Android用户不会使用过于复杂的PIN码来锁定自己的手机，对于Android默认的6位PIN码来说，任何普通的家用电脑都能在1～2小时内完成破解，而如果使用专业的大型计算机来进行攻击，可破解的PIN码范围将大大提升。在Android4.4中，Google升级了FDE所使用的密钥生成算法[4]，导致进行暴力破解密钥所花费的时间有了大幅度的提升，但是FDE本身的机制并没有变化，仍然为攻击者留下了可能的攻击窗口，过于简单的PIN仍然能够在可以接受的时间内被碰撞出来。

全磁盘加密的这些不安全特性会在磁盘数据残留问题的影响下被大大放大，对于被遗弃或意外丢失的Android设备来说，即使开启了全磁盘加密，也无法阻止攻击者利用磁盘上残留的数据来进行上述的攻击，甚至发现在一些使用了第三方recovery的设备上，由于recovery开发者的疏忽，想要格式化全磁盘加密过的磁盘都变得非常困难。

若要攻击开启了全磁盘加密的设备，首先需要将磁盘的相关分区提取出来，包括存储了加密数据的data分区和存储了crypto footer的特殊分区；之后解析crypto footer获取计算master key所必须的参数；然后使用一些暴力破解脚本对用户的PIN码进行暴力穷举，尝试使用不同的PIN码去解密data分区的第一个区块，如果解密后的数据符合ext4文件系统的标准格式，那么说明找到了正确的PIN码，之后就可以使用这个PIN码解密整个data分区，得到明文的数据。值得一提的是，磁盘上发生的数据残留是存在概率的，并不是每一个字节都会残留在磁盘上，这意味着从磁盘上提取到的残留信息可能破损的、不完整的，如果这些残缺恰好发生在data分区的头部或是保存crypot footer的部分，则关键数据的缺失将导致暴力破解无法进行。

基于上述的原因，本文认为传统的Android全磁盘加密方案无法达到其预想中的安全强度，也无法在存在数据残留的情况下保护磁盘数据的安全。

[1]Baekes M，Gerling S，HammerC，et a1．AppGuard-enfordng user requirements on android apps[C]//International Confe-rence on Tools and Algorithms for the Construction and Analysis of Systems(TACAS’2013)．Rome：Springer Berlin Heidelberg，2013：543-54

[2] Flash memory[OL]．[2016-5-17]．http://en.wikipedia.org/w/index php?title=

Flash\_memory

[3] Bugiel S，Heuser S，Sadeghi A R. Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies[C]//Usenix Conference on Security．2013 :131—146

[4]scrypt[OL].[2016-3-14].http://www.tarsnap.com/scrypt.html