文章编号：1004-4353(2017)01-0051-04

基于单分组散列函数的安卓应用程序保护方案的设计

吴晗， 李永珍
（延边大学工学院 计算机科学与技术学科 网络与信息安全研究室，吉林 延吉 133002）

摘要：针对安卓手机的安全保护机制，提出了一种基于单分组散列函数的数字签名机制和关键资源保护机制相结合的保护方案. 首先该方案通过安全文件生成器将关键资源文件的哈希值保存到安全文件中，然后使用数字签名机制保证安全文件不被篡改，最后由关键资源保护机制确保关键资源文件的完整性，其中包括对被篡改的关键资源进行纠错. 实验结果表明，本方案不仅能够有效地防止安卓应用程序被篡改，而且增加了逆向攻击的难度.

关键词：完整性确认；单分组散列函数；安卓应用保护
中图分类号：TP309.2 **文献标识码**：A

The design of Android application security protection scheme based on the single block hash fuction

WU Han， LI Yongzhen*
（*Network & Information Security Lab.*, *Dept. of Computer Science & Technology*, *College of Engineering*, *Yanbian University*, *Yanji* 133002, *China*）

Abstract: According to the security protection mechanism of Android mobile phone, a protection scheme based on single block hash function and digital signature mechanism is proposed. The first hash file generator will be the key security resource files saved to the file security value through the program, to ensure the safety of the file has not been tampered and then use the digital signature mechanism, and finally by the key resource protection mechanism to ensure the integrity of key resource files, including error correction of key resources has been tampered with. The experimental results show that the proposed scheme can not only prevent the Android application from being tampered with, but also increase the difficulty of reverse attack.

Keywords: integrity verification; single block hash function; Android application protection

0 引言

随着安卓(Android)操作系统的快速发展，其安全问题也日益凸显，例如 360 手机卫士在 2016 年第二季度累计监测到用户感染恶意程序 6 180 万人次[1]. Android 操作系统容易受到攻击的原因是由于用 Java 语言编写的 Android 程序很容易被逆向分析[2]，使攻击者容易获得程序源码并写入恶意代码，进而侵害使用者的利益. 1993 年，Cohen 等[3]提出了用于保护程序安全的代码混淆技术，将代码格式打乱，使攻击者难以阅读；但由于 Android 系统的自身原因，仍有大量的代码不能被混淆. 之后，Google 公司提供了 Android 本地开发工具包，该工具包允许开发者使用反编译难度较大的 C/C++语言编写动态链接库，并在

收稿日期：2016-12-12

* **通信作者**：李永珍(1971—)，男，博士，副教授，研究方向为网络安全、无线网络协议.

Java 层通过 JNI[4] 进行访问. 由于此方法中本地层的 C/C++程序可以在很大程度上预防应用程序被反编译而带来的危害,因此被广泛运用在应用程序的保护中. 对于完整性验证,目前安卓系统使用最为广泛的是自签名的数字签名技术[5],但该机制只能在安装时进行验证,这使得应用程序在使用过程中仍然存在着安全问题. 对此,曲乐炜等[6]提出了一种结合文件数据和内存数据的数据完整性检测方案,该方案不仅对检测典型恶意代码有较高的准确率,而且不依赖硬件,实用性较强. 郑光明等[7]提出了通过 MD5 检测文件完整性的方案,其主要思想是对敏感文件的信息摘要进行多次比对,从而判断文件是否被修改,该方法在一定程度上提高了对文件的保护性. 针对 Android 操作系统在使用过程中应用程序仍然存在安全的问题,本文设计了一种能够实时检测重要程序完整性的数字签名机制和完整性验证机制来保护安卓程序的方案,并通过实验验证了本文方案的有效性.

1 安全方案的设计

本文提出的安全保护方案包括数字签名机制和完整性验证机制,其具体过程为:首先,开发者将代码中的关键资源进行哈希运算,以此得到哈希值并将其存放到安全文件中,这一过程由安全文件生成器来完成;其次,通过数字签名机制保证安全文件不被篡改;最后,通过完整性验证机制来确保关键资源代码不被篡改. 由以上内容可知,只有当数字签名机制通过认证才会进行完整性验证,且任意环节验证不通过都将无法启动程序. 本文提出的完整性验证机制还提供了纠错功能,通过服务器向客户端派发正确的受保护资源. 另外,为了增强本文方案的抗逆向性能,数字签名机制和完整性验证机制都定义在 Native 层,因此只有通过 JNI 调用才能使用.

1.1 安全文件生成器

哈希函数因具有强抗碰撞性,使得其在验证数据的完整性方面成为目前应用最普遍的技术. 哈希函数的工作原理是:接受可变长度的消息,输出定长的哈希值,并且是不可逆行为. 本文使用单分组哈希函数计算关键资源文件的哈希值,并将其保存到安全文件当中. 在程序运行过程中,通过对比关键资源的哈希值和安全文件当中的哈希值来辨别关键资源是否被篡改. 本文所使用的安全文件生成器如图 1 所示,该工具支持 MD5、SHA1、SHA2、SBH 4 种哈希算法的运算[8]. 由于 SBH 哈希算法在处理数据时速度较快,因此本文选取 SBH 哈希算法来计算关键资源的哈希值.

图 1 安全文件生成器

1.2 数字签名机制

为解决普通数字签名只在下载时进行验证的问题,本文提出基于服务器的数字签名机制:首先服务器端对安全文件进行单分组散列函数的哈希运算,然后使用私钥对该哈希值进行加密,最后把安全文件和加密后的哈希值传到客户端. 客户端使用公钥对接收到的文件进行解密,并用相同的哈希函数对安全文件进行哈希运算,将得出哈希值与解密后所获得的哈希值进行比较,若值相同则说明文件在传输过程中未被篡改. 该机制在每次程序的启动时进行验证,以保证安全文件不被篡改. 数字签名机制的操作过程如图 2 所示.

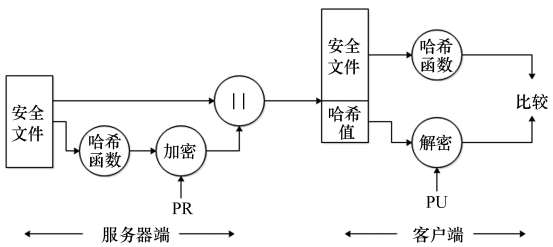

图 2 数字签名机制

1.3 完整性验证机制

程序在运行的过程中一旦加载到关键资源,就将启动完整性验证机制. 完整性验证机制计算关键资源的哈希值并与安全文件中的哈希值进行对比,若哈希值相同则说明关键资源完整,若哈希值不同,则向用户发出文件被篡改的警告;同时,程序自动从服务器端下载完整的资源文件,下载

完成后，应用程序自动重启. 在程序运行过程中，只要加载到受保护资源就需要进行验证，直到程序停止运行. 完整性验证机制的流程如图 3 所示.

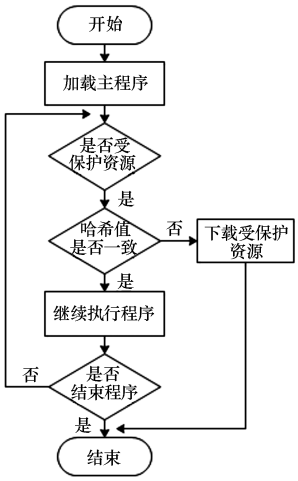

图 3　完整性验证机制流程图

2　实验测试

2.1　性能测试

为防止攻击者通过反编译手段获取到程序代码，本方案将数字签名和完整性验证都用 C++ 语言写在 Native 层，应用程序中的其他代码则使用 Java 语言写在应用层和应用框架层. 跨层调用的方法能够增加程序反编译的难度，但会在一定程度上增加时间消耗. 本方案在测试阶段使用 JNI 调用 105 次 Native 层的空函数，以此来获取跨层调用的具体耗时. 经测试，该调用仅耗时 38 ms，由此可知本方案中的跨层调用的额外开销可忽略不计. 又因为 C++ 语言效率高于 Java 语言，进一步采用相同字符串作为输入，分别在Java层和 Native 层使用 MD5 和 SBH 哈希算法对字符串做哈希运算，结果如表 1 所示. 由表 1 可知，在不同运算次数下，JNI 调用均比 Java 调用高效，说明 C++ 语言的高效性弥补了因跨层调用所产生的额外时间开销. 因此，本方案不仅能够有效防止反编译，而且还可提高程序效率.

表 1　JNI 调用和 Java 调用的性能对比　　ms

次数	MD5		SBH	
	JNI	Java	JNI	Java
500	77.47	142.13	79.60	123.56
700	104.75	194.08	113.44	163.34
1 000	147.81	285.12	155.64	235.89
2 000	309.41	635.96	324.59	567.08

2.2　实用性分析

1）数字签名机制测试. 应用程序启动后会检查客户端是否存在安全文件，若存在则比较其哈希值，值相同则提示用户验证通过，如图 4 所示. 若客户端不存在安全文件或哈希值不相同则会自动弹出下载页面，如图 5 所示. 经多次实验表明，该机制正确可行.

图 4　验证通过示意图

图 5　验证未通过示意图

2）完整性验证机制测试. 数字签名机制通过之后，安卓应用程序启动并显示主界面，如图 6 所示. 将位于该界面正上方位置的横幅“网络与信息

安全研究室”设置为关键资源,用“生物科学实验室”将其替换,如图 7 所示. 完整性验证机制检测到当前安卓应用中关键资源的哈希值与安全文件中的哈希值不同,认定其被篡改,系统自动弹出下载页面,防止关键资源不被篡改. 经多次实验表明,该机制能够保证关键资源的完整性.

图 6 关键资源完整示意图

图 7 关键资源被篡改示意图

3 结论

经实验表明,本文提出的基于单分组散列函数的安卓应用程序完整性保护方案能有效地检测出关键资源是否被篡改,并可通过服务器修正被篡改的资源来保护程序的完整性. 然而,本方案中的纠错功能只有在网络畅通的情况下才能实现,因此实现本方案的本地检测和验证是下一步的研究工作.

参考文献:

[1] 360 互联网安全中心. 2016 年第二季度中国手机安全状况报告[EB/OL]. [2016-07-29]. http://zt.360.cn/1101061855.php? dtid=1101061451&did=3701370185.

[2] 张志远,万月亮,翁越龙,等. Android 应用逆向分析方法研究[J]. 信息网络安全, 2013(6):65-68.

[3] Collberg C, Thomborson C, Low D. A taxonomy of obfuscating transformations[D]. New Zealand: The University of Auckland, 1997.

[4] 向模军. 利用 JNI 实现 Java 与 C++通信[J]. 计算机时代, 2009(12):56-57.

[5] 雷灵光,张中文,王跃武,等. Android 系统代码签名验证机制的实现及安全性分析[J]. 信息网络安全, 2012(8):61-63.

[6] 曲乐炜,罗森林,孙志鹏,等. Android 系统数据完整性检测方法研究[J]. 信息网络安全,2016(8):61-67.

[7] 郑光明,胡博. 基于 MD5 的文件完整性检测软件设计[J]. 湖南理工学院学报(自然科学版),2007,20(1):35-38.

[8] 周莹冰. 基于 S 盒的单分组哈希函数的设计与实现[D]. 延吉:延边大学,2015:9-12.