第 36 卷 第 1 期 计 算 机 工 程 2010 年 1 月
Vol.36 No.1 Computer Engineering January 2010
·软件技术与数据库· 文章编号：1000—3428(2010)01—0043—02 文献标识码：A 中图分类号：N945.15

Windows Mobile 中注册表保护的实现

吴志恩，胡劲松
(华南理工大学计算机科学与工程学院，广州 510006)

摘 要：针对 Windows Mobile 系统对钩子函数支持不足，导致不能方便地保护注册表的情况，提出为系统 API 建立钩子函数的方法。根据 Windows Mobile 系统的 API 调用过程，为操作注册表的一系列 API 建立相应的钩子函数，从而通过钩子函数有效地拦截 API。实验结果显示，该方法能有限拦截操作注册表的 API，达到有效保护注册表和提高系统的安全性的目的。

关键词：Windows Mobile 系统；注册表；钩子函数

Implementation of Registry Protection in Windows Mobile

WU Zhi-en, HU Jin-song
(College of Computer Science & Engineering, South China University of Technology, Guangzhou 510006)

【Abstract】According to the situation of Windows Mobile system which does not support normal hook functions and can not be used to protect the registry of system, this paper proposes to build hook functions for the system APIs. Based on the API call process in Windows Mobile system, it builds hook function for the API which regarding to the operation of registry, and intercepts this API. Experimental results shows that it can protect the registry effectively by intercepting the relevant API and improve the system's security.

【Key words】Windows Mobile system; registry; hook function

1 Windows Mobile系统简介

Windows Mobile 是微软基于 Windows CE 平台核心开发的适用于智能设备的操作系统[1]。Windows Mobile 系统凭借其强大的功能、优秀的用户界面，以及丰富的软件支持，在市场上的占有率日益上升。与此同时，Windows Mobile 系统的安全性也是人们关注的焦点。与 Windows 其他版本的操作系统一样，Windows Mobile 系统的注册表保存了操作系统的一些重要的启动和配置信息，对系统的安全起到十分重要的作用。所以，保护 Windows Mobile 系统的注册表，对系统的安全性会有很大的提升。

1.1 Windows Mobile系统的注册表

Windows Mobile 的注册表结构与其他版本的 Windows 操作系统注册表结构类似。注册表和文件目录树一样具有层次结构。注册表的键相当于指定路径下的目录。键中可包含更多的键或者是若干键值，键值就是注册表最终存储的数据。

Windows Mobile 系统的注册表提供了 2 种实现方式[2]，一种是基于对象存储的注册表(RAM-based registry)，其实现方式是把注册表作为一个对象存储在 RAM 中；另一种是基于 Hive 的注册表(Hive-based registry)，基于 Hive 的注册表方式使用文件存储注册表数据，这种方式使系统断电后无须备份和恢复注册表数据。无论上述哪种实现方式，对注册表进行操作都需要调用一系列进行注册表操作的系统 API。

1.2 注册表保护原理

对于 Windows 操作系统注册表的保护，一般的做法是利用钩子函数[3]截获对注册表进行操作的一组 API 函数，如 RegOpenKeyEx[4]等来实现保护。截获这类函数后，可以对函数做相应的修改，根据函数输入参数判断用户要对注册表的哪些键值做操作，从而决定是否允许用户继续其操作。这样便可以截获某些对注册表不利的操作，达到保护注册表的目的。然而在 Windows Mobile 系统中，对钩子函数的支持仅限于键盘钩子。对于其他绝大部分的 API，Windows Mobile 不提供截获 API 调用的钩子函数机制。这就使得在 Windows Mobile 实现对普通系统 API 的函数挂钩成为一大难点。而要克服这一难点，实现在 Windows Mobile 系统上利用钩子函数来保护注册表，需要在系统中建立新的钩子函数模型。

2 Windows Mobile系统钩子函数模型的实现

2.1 Windows Mobile系统API调用过程

要建立 Windows Mobile 系统的钩子函数，需要清楚 Windows Mobile 系统的 API 函数的调用过程。Windows Mobile 系统启动后，会将系统 API 函数的信息保存在相关的数据结构中[5-6]，并驻留在内存的指定地址。

如图 1 所示，PUserKData 结构保存了系统当前运行所需的相关信息，其中包括进程列表地址、API Sets 地址等。PUserKData 结构驻留在 Windows Mobile 系统的一个固定虚拟内存地址中，该结构的定义和驻留内存地址可以在相关的头文件中获得。本文所需的是该结构中的 API Sets 数组地址。一个 API Sets 是指同一个系列的 API，比如 SH_FILESYS_APIS 所指的是一系列文件系统操作所需的 API，对注册表进行操作的一系列 API 就定义在这个 API Sets 里。接下来，每个 API Sets 都用一个 CINFO 类型的数据结构来存储它的详细信息。CINFO 结构也是系统定义的数据结构，在该结构中，可以找到指向该系列 API 函数的列表指针。指针指向的是保

基金项目：国家自然科学基金资助项目(60574078)；广东省自然科学基金资助项目(31454)；广州市科技计划应用基础基金资助项目(2006J1-C0321)

作者简介：吴志恩(1984－)，男，硕士研究生，主研方向：智能算法及其应用，手机软件开发；胡劲松，副教授

收稿日期：2009-10-14 **E-mail：**wuzhien1984@126.com

存 API 函数的一个名为 Win32Methods 的数组。Win32Methods 的数组实际上保存了该系统 API 函数的入口地址。通过这个数组，最终便可以找到实际执行的 API 函数入口。Windows Mobile 系统便是通过以上的数据结构，逐步找到所需的 API 函数的入口地址来调用 API。

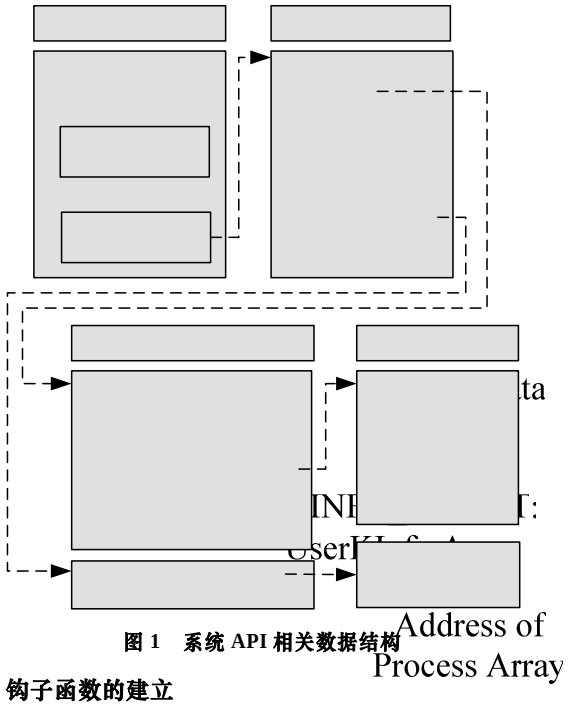

图 1　系统 API 相关数据结构

2.2　钩子函数的建立

对于上文所述的相关数据结构，要实现为 API 建立钩子函数，可以根据需要重新定义所需的 API 函数，并将指向原来 API 函数的指针，即起入口地址修改为新定义的函数的入口地址。考虑到系统的某些地址的只读操作，可行的方法是将特定的 API Sets 指向的 CINFO 结构替换成重新定义的 CINFO 结构。同时复制一份新的 Win32Methods 数组。在重新定义的 CINFO 结构中，将指向 API 函数列表的指针修改成指向新的 Win32Methods 数组，如图 2 所示。最好只需修改新的 Win32Methods 数组中的某一项函数入口地址，就可以实现为该项对应的 API 建立钩子函数。

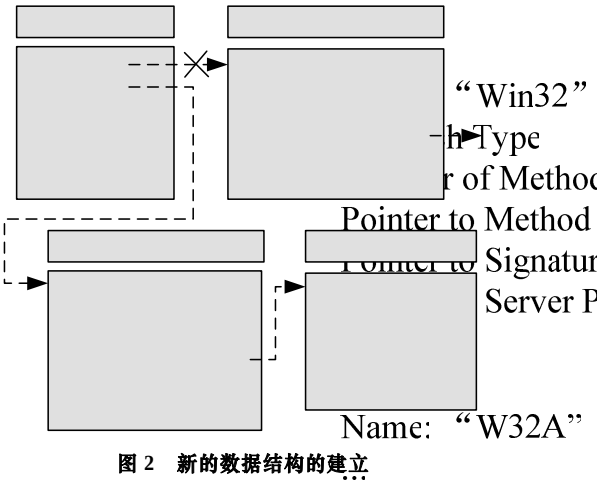

图 2　新的数据结构的建立

以对 RegOpenKeyEx API 建立钩子函数为例，RegOpenKeyEx 是 Windows Mobile 系统打开注册表所用到的 API，位于名为 SH_FILESYS_APIS 的 API Sets 中，在 Win32Methods 数组下标为 23 的元素中保存了 RegOpenKeyEx API 实现代码的入口地址。

定义新的 CINFO 结构，该 CINFO 结构与原来结构的区别是其 Methods Table 指针指向的是新的 Win32Methods 数组。而新的 Win32Methods 数组则复制了旧的 Win32Methods 数组的内容，同时将所需挂钩的 API 函数 RegOpenKeyEx 的地址，即 Win32Methods 数组下标为 23 的地址值修改为重新定义的挂钩函数地址。这样，只要将新的 CINFO 结构替换旧的结构，当用户调用 RegOpenKeyEx 这一 API 时，系统会通过新的 CINFO 结构找到重新定义的函数，从而建立 RegOpenKeyEx API 的钩子函数。这里需要注意的是，应该保存原 RegOpenKeyEx 函数的入口地址，这样就可以在建立的钩子函数中，利用该入口地址调用原来的函数来实现原有的功能。

3　Windows Mobile注册表保护的实现

3.1　注册表保护模型

对注册表的保护有多种形式，如禁止修改、删除某些注册表键值，禁止在某些路径下添加键值等。这些都可以通过对相应的 API 函数进行挂钩，或修改挂钩函数来实现。在修改的挂钩函数中，需要做的是获得函数输入的参数，通过参数判断用户使用该 API 时要对哪些键值进行操作。例如通过 RegOpenKeyEx 中的第一个参数 *hKey* 和第二个参数 *lpSubKey* 相结合，就可以获得用户想打开的注册表键所在路径。

然而这里不能通过对 2 个参数简单的判断来获得最终的路径，因为根据 RegOpenKeyEx 的函数说明，函数的第一个参数 *hKey* 的值，可以是 HKEY_LOCAL_MACHINE 等待系统预定义值，也可以是一个已经打开的注册表键值的句柄。在第一种情况下，可以通过 HKEY_LOCAL_MACHINE 等的值与参数 *lpSubKey* 的值结合来获得最终的路径。但第二种情况就不能通过 2 个参数来判断最终路径。

解决这一问题的办法是为每一个打开的注册表键值所返回的句柄与其对应的路径建立一一对应的关系，并保存在一哈希表中。因为当用户打开某一路径的键时，总会返回与这一路径相对应的句柄。通过一个哈希表，可以保存和更新句柄与路径的对应关系。当用户使用上文所述的第二种情况来调用 API 时，可以通过输入的句柄，在哈希表中查找所对应的路径。这样就可以与第二个参数 *lpSubKey* 的值结合得到最终的路径。同时，也将新的句柄与路径对应关系存入哈希表中，供后续判断所用。

在实现对操作注册表 API 函数的挂钩时，对于某些 API 可以无须建立钩子函数，以提高系统的效率。一是对注册表做只读操作的 API 函数，如 RegQueryValueEx 等；另一种是需要其他 API 打开相应路径并获得句柄，再以该句柄为参数进行调用的，如 RegDeleteValue，在使用 RegDeleteValue 前，要调用 RegOpenKeyEx 等的函数获得指定路径的句柄，以该句柄为参数来调用 RegDeleteValue。因此，可以在 RegOpenKeyEx 中就截获该操作。如果用户需要打开一些受保护的路径下的键值，在 RegOpenKeyEx 函数中就可以截获该操作，使函数打开注册表时返回失败代码。这样用户就无法调用 RegDeleteValue 函数，从而保护指定键值不被删除。

3.2　注册表保护的实现

本文根据上述的设计与实现方法，在 Windows Mobile 系统中编写了相关的钩子函数，并成功地被系统载入。具体的实现是挂接了 RegOpenKeyEx, RegCreateKeyEx, RegDeleteKey, RegSetValueEx 这 4 个 API，基本上满足了保

（下转第 48 页）