

Windows Mobile 的智能终端上内存泄露检测研究

李伟^{1,2}, 柳长安¹, 芦东昕^{1,2}, 徐立峰³

(1. 华北电力大学 计算机科学与技术系, 北京 102206;

2. 中兴软件技术有限公司成都研究所, 四川 成都 610041;

3. 中兴通讯股份有限公司成都研究所, 四川 成都 610041)

摘要: Windows Mobile 5.0 是微软为智能移动终端推出的软件平台, 特别在智能手机领域受到越来越广泛的应用; 在基于 Windows Mobile 的产品设计中, 内存泄露又是需要考虑的关键之一。文中解析了 Windows Mobile 平台上监测设备内存泄露的工具 AppVerifier, 并在模拟器上实现了监测应用程序内存泄露, 进而发现使用 AppVerifier 存在的问题, 并指出了使用中高效利用 AppVerifier 的方法。

关键词: Windows CE; Windows Mobile; 内存泄露检测; AppVerifier

中图分类号: TP309

文献标识码: A

文章编号: 1673-629X(2006)11-0109-03

Research on Detecting Memory Leak in Windows Mobile - Based Device

LI Wei^{1,2}, LIU Chang-an¹, LU Dong-xin^{1,2}, XU Li-feng³

(1. Dept. of Computer Sci. & Tech., North China Electric Power Univ., Beijing 102206, China;

2. Chengdu Institute of ZTE Software Corporation, Chengdu 610041, China;

3. Chengdu Institute of ZTE Corporation, Chengdu 610041, China)

Abstract: As a software platform, Windows Mobile 5.0 is Microsoft production launched in intelligent mobile terminals and is being widely used especially in smartphone or pocket PC. On the other hand, memory leak is a key point for Windows Mobile-based device development. Discussed how AppVerifier, a tool employed in detecting memory leak in Windows Mobile, works in detail. And introduced AppVerifier in the device emulator as a case in point. Specified a problem existing in AppVerifier and a method in using the tool more efficiently.

Key words: Windows CE; Windows Mobile; memory leak detecting; AppVerifier

0 引言

内存泄露是指系统中存在未回收的内存^[1]; 内存泄露会逐渐消耗越来越多的内存, 最终导致内存不足或系统崩溃。在嵌入式移动终端中, 内存一般比桌面计算机的内存小得多, 如果存在内存泄露, 设备很容易出现故障。有效地避免内存泄露对于提高嵌入式移动终端的稳定性起着非常重要的作用^[2]。

Windows Mobile 是 Microsoft 用于 Pocket PC 和 Smartphone 的软件平台, 面向个人移动电子消费市场, 最新版本是 Windows Mobile 2005, 现在 Windows Mobile 的合作伙伴已扩展到全球 48 个国家的 40 家硬件厂商和 68 家移动运营商。对原始设备制造商(OEM)来讲, Windows Mobile 是一个集合, 包括一组事先定制的 Windows CE(微软的通用嵌入式操作系统)组件、其他附加的与 Windows Mobile 相关的组件和 OEM 所必需遵守的产品功能性要

求。OEM 利用这些组件, 并添加必要的设备驱动程序和其他应用, 然后烧到设备的 ROM 中。针对如何检测 Windows Mobile 上设备运行应用程序时可能出现的内存泄露, 文中分析了 Application Verifier 的工作原理, 并以 Windows Mobile 的模拟器 Device Emulator 作为目标设备, 检测模拟器上运行应用程序出现的内存泄漏; 进而指出了实际应用中可能遇到的问题及相应解决方案。

1 Application Verifier 的工作原理

1.1 内存泄露的原因

在嵌入式移动终端中, 内存泄露的主要原因是^[3]:

- * 内存块分配之后没有进行相应的回收工作;
- * 程序代码设计有问题, 导致内存块无法回收。

实际情况是, 在应用程序中如果忘记关掉句柄或及时销毁 Graphics Device Interface (GDI) 对象等意外情况发生时, 则会产生内存泄露, 进而影响到整个系统的性能^[4]。优秀的程序员也有可能无意识中在设计应用程序时犯了内存泄露的错误。

1.2 Application Verifier 的工作原理

为了减少并避免内存泄露及其它形式的资源泄露问

收稿日期: 2006-01-18

作者简介: 李伟(1981-), 男, 山东临沂人, 硕士研究生, 研究方向为嵌入式操作系统、第三代移动通讯; 柳长安, 博士, 副教授, 研究方向为嵌入式系统、机器人技术、虚拟现实。

题,微软开发了 Application Verifier (AppVerifier)。AppVerifier 在应用程序运行时,同时进行动态监测各种形式的资源泄露,如内存泄露、句柄泄露和 GDI 对象泄露等。因此,AppVerifier 可以发现目标设备上应用程序设计的错误,提高设备的稳定性。运用 AppVerifier 可以降低 Windows Mobile 设备上应用程序查错的难度。

AppVerifier 运行的每个测试叫做一个插桩程序^[5],实际上是一组特殊的动态链接库(DLL),AppVerifier 还使用了 Windows CE 中的插桩程序引擎,该引擎可以看作是 Windows CE 内核的延伸。插桩程序是同内核加载器一起工作的。内核加载器负责加载和解析应用程序的动态链接。当有 .exe 文件或者 .dll 文件被加载后,加载器会通知插桩程序引擎已加载的内容,然后插桩程序引擎会查看注册表以判断是否对加载的 .exe 文件或者 .dll 文件进行插装。也就是说,插桩程序引擎在应用程序和操作系统之间插装插桩程序,这样,插桩程序就被装载到进程空间,当有调用发生时,内核就直接调用插桩程序而不是直接从其他库中调用了,实现了插桩程序向操作系统解释函数调用。

其中,注册表中插桩程序的信息在[HKEY_LOCAL_MACHINE] \ shim engine 目录下。每个注册键下的子键列出了待插装的应用程序。每个子键下是要插装的应用程序及需要插入的向操作系统解释调用的 DLL—插桩程序。

图 1 是已经插入了插桩程序的名字为 GDILeak 应用程序的设备注册表的信息。

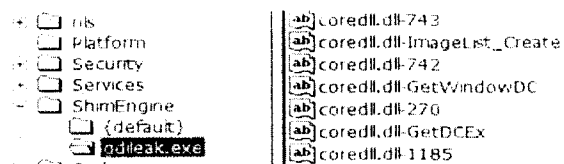


图 1 插桩程序引擎的注册表信息

如果注册表信息显示应该对 .dll 文件或 .exe 文件进行插装,插桩程序引擎将插桩程序 DLL 作为调用函数的输入。当该函数调用相关的 API 函数时,实际上是转向调用了插桩程序 DLL。因此插桩程序也就实现了向操作系统传递调用。正因为 AppVerifier 提供了大量的插桩程序,才提供对插桩程序引擎的支持。当 AppVerifier 与远程设备连接后,AppVerifier 会下载 DLL 到设备上。用户在目标设备的主界面上点击 Add 按钮添加 .exe 文件或者 .dll 文件时,AppVerifier 就修改注册表中的信息,将最新的应用程序添加到注册表,并添加数值将对分配和释放句柄、GDI 对象和内存块的调用映射到对插桩程序的调用。之后插桩程序跟踪具体的分配和释放,如果有分配和释放不匹配的情况,就会报告可能有资源泄露。从而实现了对内存等资源的监视。

2 利用 AppVerifier 实现检测应用程序内存泄露

使用 AppVerifier,首先启动 AppVerifier,对应的应用

程序名字为 appverifce.exe,一般在默认路径 C:\Program Files\Platform Builder for Windows Mobile\5.00\CEPB\WCETK\DDTK\DESKTOP;也可以直接从 Windows CE Test Kit 工具中启动。其次,要建立 AppVerifier 与目标设备的连接。建立好连接之后就可以选择并运行需要监视的应用程序了^[6]。运行结束之后就可以分析 AppVerifier 得出的结果了。

以如下的代码进行分析:

```
// 等待远程连接请求
while (fContinue) {
    LogIt (hWnd, TEXT("waiting..."));
    nSize = sizeof (t_btaddr);
    // Block on accept
    r_sock = accept (s_sock, (struct sockaddr *)&t_btaddr, &nSize);
    if (r_sock == INVALID_SOCKET) {
        LogIt (hWnd, TEXT("accept failed %d"), GetLastError());
        break;
    }
    LogIt (hWnd, TEXT("sock accept..."));
    CreateThread (NULL, 0, ReceiveThread, (PVOID)r_sock, 0, NULL);
}
closesocket (s_sock);
```

在上面的应用中,当客户端每次与 TCP/IP Socket 连

接时,都会创建一个线程,但却漏掉了一个句柄。这样,随着该应用程序的使用,连接 TCP/IP Socket 的次数不断增多,漏掉的句柄也在增加,内存泄露就发生了,最终可能导致该系统内存不足甚至系统崩溃。

运用 AppVerifier,可以很容易地发现这个应用程序的漏洞,从而检测内存泄露源。该应用也说明了另一个问题,使用 AppVerifier 时,应该尽可能地发现应用程序中所有的问题,在上述代码中,如果不使客户端使用 TCP/IP Socket 与应用程序连接,就不会发现所存在的问题了。

3 应用中发现的问题及其对策

使用 AppVerifier 过程中,可以发现在结果中记录的相关信息要比实际有效信息要多。产生该问题的原因是:一种情况下 AppVerifier 可以判断出与所检测到的内存泄露相关的函数调用等行为,但无法判断哪种是内存泄露的真正原因。因此,为了保证信息的完整性,AppVerifier 将所有相关的信息全部保存到记录的日志中,而让开发人员去分析出真正的原因;而另外一种情况则可能是当应用程序开发者在编写程序的时候没有释放句柄,但这并不会导致内存泄漏,因为其应用程序在结束之前会有释放所占用的资源的机制,该情况下利用 AppVerifier 检测该程序运行时,AppVerifier 是不会判断在应用程序中未释放的资源

是合法的,在 AppVerifier 所记录的日志里,也包含了程序运行过程中资源的分配与释放的详细信息,于是日志里的这种信息就多了。下面是经典的“Hello,World”程序代码:

```
// Hello.cpp : 定义了控制台应用程序的入口点
#include "stdafx.h"
#include <windows.h>
#include <comctl.h>
int tmain(int argc, _TCHAR * argv[])
{
    return 0;
}
```

图 2 是 AppVerifier 分析该应用程序代码后得到的日志信息,不过该日志信息是错误的。

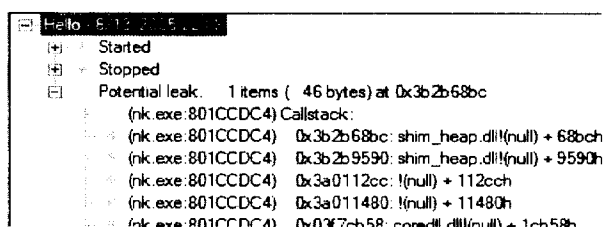


图 2 记录了错误信息的日志报告

很明显,“Hello,World”是个经典程序,不会存在内存泄露的问题。但是日志中却记录了内存泄露信息,说明 AppVerifier 的分析有问题。这就是上述原因的一个实例。因此,在分析其他应用程序的测试日志时,应该意识到这种假象。为了说明另一种潜在的内存泄露,可以在“Hello,World”应用程序中增加一行,创建一个消息框,结果是 AppVerifier 测试后的日志记录了更多的这种无关信息。示例代码如下:

```
// Hello.cpp: 定义了控制台应用程序的入口点。
#include "stdafx.h"
#include <windows.h>
#include <comctl.h>
int _tmain(int argc, _TCHAR * argv[])
{
    MessageBox (NULL, TEXT("Hello World"), TEXT
("Title"), MB_OK);
    return 0;
}
```

分析图 3,测试日志中提示的内存泄漏问题,可以发现是由 Windows Mobile 设备的输入管理器引起的。事实上,对 Windows Mobile 5.0 中的应用程序进行 AppVerifier 测试,只要应用中创建了窗口,测试日志就会记录下类似的信息。最好的方法就是对测试日志提供的信息进行仔细分析,对函数调用过程进行跟踪,从而得出有用的信息,找出是不是确实存在内存泄漏问题,如果存在,进一步确定导致内存泄露的原因。

4 结束语

AppVerifier 是一种有用的工具,可以有效跟踪嵌入式

移动终端上应用程序中可能存在的各种形式的资源泄漏问题,如句柄泄漏、GDI 对象泄漏等,并提供了良好的测试界面^[7]。该工具对于嵌入式设备或移动终端开发者是很有益的。此外,AppVerifier 还支持远程操作,可以运行在 Windows Mobile 的设备上,将目标设备和开发工作站建立连接后,也可以运行在开发工作站上的 Platform Builder 环境下。如果目标设备没有友好的用户界面或者根本没有用户界面,则 AppVerifier 支持远程操作的功能可以大大地方便检测目标设备上的应用程序,以保证所开发的应用的稳定性。

Log results:

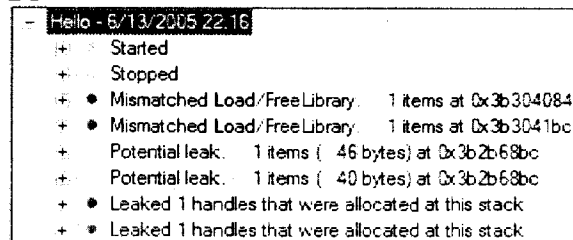


图 3 当有线程创建新窗口式 AppVerifier 在日志中错误记录的信息

在实际应用中,一方面 AppVerifier 为监测 Windows Mobile 的设备应用程序内存泄露等方面提供了极大方便,另一方面,由于 AppVerifier 本身的特点,还需要注意几个问题:AppVerifier 提供的测试日志中信息量较大,是无效信息和有效信息的集合,因此需要清楚日志中每条信息的原因,从而定位资源泄露的位置;同时在设备上使用插桩程序在一定程度上会影响设备的性能,若操作不当,关掉 AppVerifier 之后插桩程序注册表数据仍然会存留在设备注册表中。尽管 AppVerifier 存在一些问题,但在检验 Windows Mobile 上的设备应用程序方面仍是很有帮助的,否则单对应用程序的代码进行测试要困难得多。

参考文献:

- [1] 徐 文. 一个动态内存管理模块的实现[J]. 单片机与嵌入式系统, 2002(9): 26-28.
- [2] 吴 民. Linux 下面向函数的动态内存泄漏监测[J]. 计算机工程与应用, 2003, 39(6): 37-40.
- [3] Mohamod A B. Using genetic algorithm to improve the performance of multi-host vulnerability checkers[J]. IEEE. 2001. 68(10): 92-97.
- [4] 王泽民, 芦东昕, 徐立峰, 等. 嵌入式系统软件内存泄漏监测的算法和实现[J]. 计算机工程, 2005, 31(13): 84-86.
- [5] 周振喜, 戴国骏, 陈晓峰, 等. Windows 应用程序移植到 Windows CE 下的策略[J]. 计算机工程与设计, 2004, 25(9): 35-37.
- [6] 何华灿, 李 新. 单窗口嵌入式 GUI 的页面管理研究[J]. 计算机应用研究, 2002, 19(6): 35-37.
- [7] Microsoft Co. Ltd. Windows Mobile version 5.0 Help Documentation[Z]. 2005.