LEAKPOINT**: Pinpointing the Causes of Memory Leaks**

**周昊一 MF1233055**

内存泄漏是一种非计划中的内存消耗,会对程序的性能,甚至是正确性带来很大的影响.在用C或是C++写的程序中,通过使用malloc或是new来手动分配内存,这两个函数会申请一块空闲的内存空间m,并将一个指向m首地址的指针p返回.当m不再需要被使用的时候,需要调用free或是delete函数,将这块空间收回.如果m没有在适当的时刻被回收,内存泄漏发生了.内存泄漏有两种,一是lost memory,指一块内存空间m没有被释放,但是再也无法访问到;第二种是forgotten memory,值一块内存空间m没有被释放,也可以被访问到,但是在每一次访问后既没有被释放,也没有再被访问.

内存泄漏问题是很难被检测到的,因为它们不会立即就引起错误,而只有在当程序消耗了大量的内存时,才会被察觉到.当一个有内存泄漏问题的程序运行在系统上时,它还会因为占用大量内存而影响到其他程序的性能.并且即使是在成熟的程序中,内存泄漏也是一个很常见的问题.

正是由于内存泄漏导致的这些问题以及它的常见性,许多研究人员提出了各种静态或是动态的方法来检测它.而本文则是在已有工作的基础上提出了一种新的内存泄漏检测技术,不仅能提供与现有方法一样多的信息,而且能够指出发生内存泄漏的位置.

本文所使用的方法基于一种叫dynamic tainting的技术,这种技术在程序执行时记录程序的有关信息.过程大概分为三步:1.用污点标记感兴趣的程序数据(如变量,内存地址);2.在被标记的数据被访问时对污点进行增殖;2.检查在运行过程中一个特定的点上与其有关的污点.

对于每一个污点需要记录5个信息:1.进行内存分配的位置;2.分配的内存大小;3.是否被释放了;4.指针计数;5.最后一次引用的位置.

在污点的更新和增殖过程中,进行两种检查来发现内存泄漏:1.当一个被tm污染的指针丢失或者被覆盖的时候,检查它是不是到m的最后一个引用,以此可以发现lost memory;2.对于一块没有被释放的内存m,总是报告它最后一次被引用的位置,以此来发现forgotten memory.

为了进行tainting操作,需要对系统的所有涉及内存申请和释放的操作进行截取(如new, delete).除此之外,为了支持memory pool中的自定义内存管理,还允许开发者额外指定内存相关操作.

propagation操作则是需要截取赋值,加,减,与这些操作.