第 35 卷 第 8 期 计 算 机 工 程 2009 年 4 月
Vol.35 No.8 Computer Engineering April 2009
·软件技术与数据库· 文章编号：1000—3428(2009)08—0116—02 文献标识码：A 中图分类号：TP311.5

基于虚拟技术的操作系统内存崩溃修复方法

孙 通
(复旦大学平行处理研究所，上海 201203)

摘 要：提出一种基于虚拟技术的操作系统内存崩溃修复方法，该方法解决远程修复技术对网络的依赖问题，同时克服本地修复存在的局限性。依据该方法实现的原型系统能在较短时间内对崩溃的系统内存进行修复，实验结果表明，添加修复功能的系统所产生的性能损失是可以接受的。

关键词：虚拟技术；操作系统；内存崩溃

Recovery Method for Memory-crash of Operating System Based on Virtual Technology

SUN Tong
(Institute of Parallel Processing, Fudan University, Shanghai 201203)

【Abstract】A recovery method for memory-crash of operating system based on virtual technology is proposed, which solves the problem of reliability of network for remote repairing, and gets over the localization of local recovery. The recovery system using this method can repair the crashed memory in a short time. Experimental results show the loss of performances in the system with this new function of reparing is acceptable.

【Key words】virtual technology; Operating System(OS); memory-crash

1 概述

目前，计算机在各种关键领域内得到广泛应用，计算机系统的可靠性和容错能力日益成为系统设计上需要重点考虑的问题。本文提出一种基于虚拟技术的操作系统内存崩溃修复方法，实现 XF 修复系统，从而有效提高当前通用操作系统的可靠性和容错能力。

恶意进程的破坏是导致系统崩溃的主要原因之一，这些恶意进程往往通过占用大量系统内存，削弱系统的处理能力，从而达到攻击操作系统的目的。因此，操作系统应具有抵御被这类恶意进程破坏的能力。

现在已有一些关于操作系统内存崩溃修复的研究，主流的解决方案是远程机器修复[1]和系统的本地修复[2]。但这 2 种方法仍然存在不足。远程修复系统依赖于网络，本地修复系统的解决手段有局限性。因此，本文利用虚拟化技术解决这个问题。

2 背景知识和相关工作

2.1 背景知识

图 1 是系统虚拟化的基本结构[3]。

图 1 系统虚拟化结构

在底层硬件与运行其上的操作系统间插入一个中间层，通常称为虚拟机监控器(Virtual Machine Monitor, VMM)。虚拟机监控器管理底层的硬件资源，负责将这些资源分配给上层运行的操作系统。由于上层操作系统的状态、资源和执行完全受控于虚拟机监控器，因此可以方便地对操作系统进行动态监控和修复。

操作系统中的恶意进程通常被赋予过大的权限，这些权限会被滥用而危及操作系统的安全。利用虚拟机监控器可截获运行其上的操作系统的系统调用，同时监控进程执行，并对恶意进程进行适当的处理，最终达到修复操作系统的目的，从而提高系统可靠性和容错能力。

2.2 相关工作

Backdoor[1]系统是由美国马里兰大学提出并实现的一个对操作系统进行远程修复的方法。该方法的所有监控和修复工作都是由远程计算机完成的，远端机器通过网络获得本地机操作系统的内存等系统信息，并通过网络对出现问题的操作系统进行修复。远程监控固然有很多优点，但其最大弊端在于网络的不可靠性，一旦网络出现问题，整个修复系统就会瘫痪。

Recovery Box(RB)[2]采用本地修复的策略，系统开辟一块固定的内存用来储存系统状态，这段内存是独立的，其他应用程序不可访问。当系统状态出现异常，修复系统会重新启动操作系统，并使用 RB 中保存的状态恢复系统。RB 能够正常恢复系统状态的前提是开辟的独立内存不受侵害，但是如果整个操作系统受破坏独立内存受到影响，那么“重启方式”也无法正确恢复系统状态，而且由于 RB 系统中保存的系统状态并非即时状态，只能将操作系统恢复到最近的一个保存点，因此可能造成部分系统状态的丢失。

作者简介：孙 通(1983－)，女，硕士，主研方向：系统软件
收稿日期：2008-11-10 **E-mail：**suntong@fudan.edu.cn

3 系统架构

3.1 系统机制

操作系统内存崩溃的主要原因是：恶意进程不符合正常规律的创建进程导致内存被非法占用，正常执行的进程无法运行。以交通流量管理为例，如果所有的车辆都遵守规则行驶，即使车辆再多也仅仅是车流相对缓慢，不会造成整个交通系统瘫痪，但如果有些车辆违反交通规则，必将影响其他正常行驶的车流，并且造成交通瘫痪，操作系统亦是如此，为保证操作系统正常可靠地运转，修复系统需要识别出这些恶意进程，并对它们进行适当的处理，恢复系统的正常运行，同时保证其他正常进程不受到影响。

然而现代操作系统本身具有极高的复杂性，这给具体的实现造成很大困难，而利用系统虚拟化技术的特点，XF 修复系统能够很好地克服这些困难。

首先是如何监控操作系统中的进程并获取它们的状态信息。XF 修复系统使用虚拟机监控器来截获进程产生的特定系统调用，依据装载到 CR3 中的页表信息判别操作系统中的进程，并记录进程的创建信息来建立所有进程的父子关系，这样能够很好地实现对操作系统中各进程的监控。其次，操作系统内存被耗尽失去处理能力，此时如何进行修复操作是另一个困难。由于虚拟机监控器使用与操作系统完全隔离的内存空间，因此不会受到操作系统内存耗尽的影响。以上 2 点正是利用系统虚拟化技术进行修复的优势所在。

3.2 系统架构

图 2 给出了 XF 修复系统的总体架构。

图 2 XF 修复系统架构

XF 原型系统在整体架构上分成 3 层：用户接口层，用户级别监控层和系统级别监控层。它们依次位于用户空间、操作系统空间和虚拟机监控器空间。相对独立的设计使整个架构清晰，且更易于实现和维护。

(1)用户接口层：XF 系统在用户空间开发一个用户态程序作为接收用户信息、进行用户设置的接口。下文提到监控程序中判定恶意进程的计数器阈值，都可以通过用户层接口程序设置。

(2)用户级别监控层：以可装载内核模块的形式被插入到操作系统中，用以获取每个进程使用内存的情况，检测超过正常使用内存的进程，并初步筛选恶意进程。

(3)系统级别监控层：位于虚拟机监控器中。监控程序负责监控操作系统中的所有进程，根据进程间的相互关系划分进程，找到恶意进程并对其执行适当修复操作，恢复操作系统的正常运行。

4 实现

4.1 监控进程

用户级别的监控程序是利用 vm_struct(管理虚拟内存的结构)中的信息来监控进程的，vm_struct 的数据结构如下，其中，size 是进程所使用的内存大小。通过对系统正常运行的进程所占内存大小进行评估，确定一个相对合理的阈值。一旦恶意进程所占用的内存值超过阈值，就会触发用户级别监控程序执行修复操作，释放内存恢复系统状态。

```
struct vm_struct{
  unsigned long flags;
  /*virtual memory block state flag*/
  void * addr;
  /*pointer to the virtual memory block*/
  unsigned long size; /*block size*/
  struct vm_struct * next;
/*pointer to the next block*/
};
```

然而，用户级别的监控具有局限性，本系统更关注于对操作系统级别的监控。使用虚拟机监控器对操作系统进行监控，当操作系统启动时，运行在虚拟机监控器中的系统级别监控程序随之启动。这个监控程序负责监控所有进程，用来判定进程间的相互关系，找到恶意进程。要做到成功地监控进程，需要追踪每一个进程的创建和终止，在虚拟机监控器中建立对应的数据结构，同时还需要追踪进程间的上下文切换以及记录不同进程的系统状态信息。

目前，系统级别监控程序使用装载到 CR3 中的进程页表信息来判别进程，并通过对 sys_fork 系统调用的截获，记录进程间的相互关系，同时也记录进程对系统资源的使用情况。比如，监控程序截获创建进程的 sys_fork，获取每个系统调用的进程号，即父进程的进程号；同时也记录新创建进程的进程号，即子进程的进程号。于是将全部进程根据所属的父进程分类，每个父进程以及它创建出来的子进程就构成了一个进程组。与此同时，监控程序为操作系统中每个进程分配一个计数器，每产生一个子进程，计数器就相应加一。

在一段监控时间内，统计进程对应的计数器值，并计算在这段时间内产生子进程的速率。通过子进程数和创建子进程的速率来判定进程是否为恶意进程，当计数器值或者速率超过阈值时，监控程序执行相应的修复操作。

4.2 修复过程

用户级别修复程序位于操作系统中，能够通过直接调用“系统调用”(system call)来终止进程，而位于虚拟机监控器中的系统监控程序没有现成的方法来直接终止位于操作系统中的进程。因此，XF 修复系统通过制造一个伪系统调用的方法来实现，进程的正常退出是通过调用 sys_exit 系统调用。在系统级别修复程序中，将 EAX(存放系统调用号的寄存器)里面的返回值填上 sys_exit 对应的系统调用号，实现伪系统调用。如果操作系统中有多个恶意进程，可以通过多次调用，将恶意进程全部终止，恢复操作系统的处理能力。

5 实验与分析

XF 原型系统的实现基于版本为 3.0.2 的虚拟机监控器 Xen，内核版本为 2.6.16 的 Linux 操作系统。测试的硬件环境为：3.0 GHz 的 Pentium 4 处理器，1 GB 内存，Intel Pro 100/1 000 以太网卡和 250 GB 7200 RPM SATA 硬盘。

5.1 正确性分析

以下是用于正确性测试的恶意进程：

```
void main( ) {
        for (;;) fork( );
}
```

(下转第 121 页)