文章编号:1001 - 2486(2005)06 - 0030 - 05

用错误围堵策略建立可生存的网络安全设备内核*

杜　皎[1,2],李国辉[1],冯登国[2]

(1.国防科技大学 信息系统与管理学院,湖南 长沙　410073;
2.中国科学院研究生院 信息安全国家重点实验室,北京　100049)

摘　要:提出在 IBM 虚拟机器的架构上,使用错误围堵策略建立可生存的网络安全设备内核的思想。建立有效的资源管理器,分割、调度机器资源,把物理资源提供给虚拟机器,平衡错误围堵与其它的性能要求。利用软件和硬件错误围堵技术制约对系统攻击引起的错误,防止一个错误引起整个系统的崩溃。用以上策略建立了内核模型,给它加上大负荷,正常和异常的工作。实验结果显示:即使在系统中某些部分出错的情况下,依然不影响系统的整体性能,并且错误围堵的开销几乎可以忽略不计。

关键词:可生存技术;错误围堵;虚拟机器;网络安全设备;内核

中图分类号:TP393.08　　文献标识码:A

Utilizing Fault Containment to Construct a Survivable Network Security Device Kernel

DU Jiao[1,2],LI Guo-hui[1],FENG Deng-guo[2]

(1.College of Information System and Management, National Univ. of Defense Technology, Changsha 410073,China;
2.State Key Laboratory of Information Security, Graduate School of Chinese Academy of Sciences, Beijing 100039,China)

Abstract: Fault containment is proposed to construct a survivable kernel of the network security device based on the IBM virtual machine. This is accomplished by setting up an efficient resource manager to supply physical resources to the virtual machine and to balance other performance requirements. Software and hardware fault containment technology is used to protect against system attacks, and avoid a system breakdown from a single fault. Model and tests prove this idea and the overheads are almost negligible.

Key words: survivability techniques; fault containment; virtual machine; network security device; kernel

随着网络技术的迅猛发展,网络已经和每个人息息相关。为了保证人们在网络上的充分自由与合法权利,网络安全更成为网络不可分割的部分。大多数的网络安全事故都是因为没有正确安装或设置网络安全设备而引起的,网络安全设备是网络安全的基础设施。但是,即使装上了这些技术先进、功能完善的网络安全设备,系统也不能确保安全。事实上,有的黑客攻击根本就是面对网络安全设备本身的,他们利用安全设备内操作系统本身的安全漏洞,绕过了现有的防火墙机制。

从另一方面来说,现存的各种操作系统并不是专门为安全设备设计的,它是一个能完成复杂功能的大管家,对于专门的功能来说,就显得结构复杂,数据量庞大,易受其它与安全不相干事件的干扰;针对真正需要实现的功能却要求通过好几层抽象,系统开销大,资源利用不充分,效率很低。另外,系统中一个错误可能会引起整个系统的崩溃。所以,迫切需要建立高效稳固的网络安全设备平台,专门用于网络安全设备。安全设备需建立在某种操作系统之上,操作系统的选择和安全设备的实现是紧密联系的。只有为安全设备平台建立稳固的内核,才能在遭遇各种面向底层的攻击时,依然维持网络安全系统的正常功能。

1　系统设计目标

我们的系统设计目标是:建立稳固高效的安全设备平台内核,为各种网络安全设备提供良好的开发

* 收稿日期:2005 - 04 - 10
基金项目:国家 863 高技术资助项目(2003AA144050)
作者简介:杜皎(1972—),女,工程师,博士生。

和运行环境:在安全设备平台本身遭受攻击时,依然维持系统的正常功能,或者做到系统性能的平稳降级。

为了提供模块化结构和高性能环境,网络安全设备的操作系统内核应该只包含最少量的功能模块。现有的操作系统很大,效率低,适应性差,不适用于网络安全设备平台。实际上,大多数操作系统的性能和适应性问题都可以通过降低操作系统的接口来解决。这一方面是通过最大程度上的机制和策略的分离,另一方面是通过提供简单的抽象来实现的。该内核的特点是:

· 采用可生存技术,使得系统在受到攻击时,仍然能维持正常功能;

· 采用错误围堵技术,把系统中错误的影响限制在很小的范围内,不影响系统的其它大部分功能;

· 以 IBM 的虚拟机概念为基础。

2　技术背景

2.1　可生存技术

不论是计算机网络系统还是数据库系统,包括计算机系统本身,都面临着越来越多的攻击,因此带来极大的安全隐患。而现有的防御机制不能抵抗所有的攻击,总有一些攻击能够成功进入要保护的系统。那么,如何处理这些成功的攻击?虽然,一些检测技术能通知你哪里出现问题,但是,许多攻击能绕过身份认证和访问控制技术。所以,需要一种技术来帮助我们在受到攻击后依然能够维持系统的正常功能,这样的技术就叫做可生存技术。使用这样的技术来进行安全防御并维持系统正常功能的系统,叫做可生存系统[1,2]。

目前对生存技术的研究方法主要有两类方法:可生存设计和入侵响应[3,4]。可生存设计需要重新设计计算机系统,使得该生存设计能够在被攻击的时候依然正确地运行。入侵响应不用重新设计系统,而是给系统装备入侵响应设备,在它的帮助下,能够在入侵来临时依然维持系统的正常运行,不会拒绝服务或使服务失真。

人们更多使用第二类研究方法入侵响应,它的开发费用低,周期短;但是系统运行时,为了获得可生存能力的开销很大。相比之下,第一类研究方法可生存设计需要重新设计系统,需要较高的费用和较长的开发周期,但系统运行时的可生存方面的开销几乎可以忽略不计。为了缩短周期,节约费用,我们把内核建立在 IBM 虚拟机器概念的基础上。

2.2　虚拟机器

虚拟机器的概念是 IBM 提出的。它的目的是对新发布的操作系统进行检测。后来,VAX VMM 安全内核[8]使用虚拟机器的概念[9]对 Intel 的 X86 进行了成功的虚拟,表明不需要对操作系统本身进行任何修改,就可以对整个硬件进行虚拟。

我们使用虚拟机器提供的通用性和灵活性来屏蔽对系统资源的处理,开销非常小。维护虚拟机器需要一个资源管理器,对资源访问模式有良好的规划。我们在虚拟机器之上建立虚拟机监控器,对系统资源进行合理划分和调配,在支持错误围堵策略的同时获得各方面性能的平衡。

3　错误围堵技术设计内核

各种各样的原因都会导致错误的发生,系统必须能够进行有效处理。假设每个系统成员都可能存在潜在的错误,那么系统错误发生的可能性和系统成员数量成正比。所以在传统的系统中,系统中成员数量越多,则发生错误的可能性就越高。在不能及时察觉错误的系统中,一个错误会导致整个系统的崩溃。即使某些任务没有使用出错的资源,也会受到影响。因此,传统的系统设计使得一个任务受系统错误影响的概率和系统中成员的数量成正比,而不是该任务所使用资源的数目。

错误容忍技术已经为大家所熟知,它帮助系统在发生错误时依然维持任务的运行。虽然错误容忍技术被证明是有效的,但是它要求复制资源,为每个系统成员和检查点保留一个备份,保存主要成员的状态信息,避免恢复时过长时间的延迟,所以开销很大。错误围堵技术[6],就是在错误发生后,把错误所

使用的资源和产生的影响与系统的其它部分隔离开。相比较而言,错误围堵技术开销较少。它把错误带来的影响限制在很小的范围内,而系统的其它部分依然能够维持正常的运行。和错误容忍技术不同的是,如果系统支持错误围堵技术,一些运算结果会因为系统出错而丢失,但是,系统的大部分不会受到该错误的影响,而且运行时的开销几乎可以忽略不计。

3.1 一种错误围堵内核结构

我们设计开发了一个内核模型,它建立在 IBM 的虚拟机器基础上,这样可以省去大量的修改操作系统所需付出的开销。为了能在虚拟机器上顺利运行,该模型对各硬件成员进行虚拟,包括进程、内存和 I/O 设备。我们还建立了资源管理器来对系统各项资源进行有效管理。

软件错误围堵对虚拟机器环境来说有直接的好处。因为系统中的监控器可以很容易地为每个虚拟机限定资源,系统被分成多个单元,错误产生的影响被限制在发生错误的单元内。只有使用出错单元内资源的虚拟机器才会受到影响,其它的虚拟机器则保持正常状态。其它单元为了能在出错时保持生存,每个单元为自己的数据保留一个备份。错误围堵要求资源管理器知道围堵的边界,因为如果虚拟机器向每个单元都提供所有资源的话,就会变得易受攻击。虚拟机可以不受其它虚拟机中软件错误的影响,因为每个虚拟机只在自己的虚拟环境中运行,其它虚拟机不能进入。系统内核如图 1 所示。

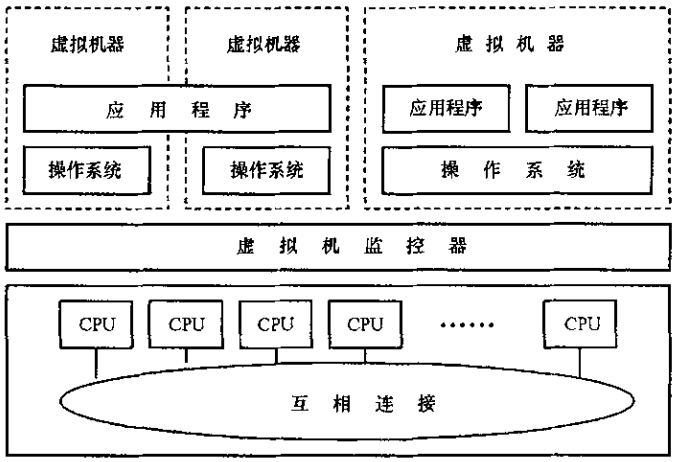

图 1 系统内核基本结构

Fig.1 Framework of system kernel

如文献[10,11]中描述,光靠软件技术本身不能提供硬件错误围堵,还需要硬件的合作。通过和硬件的合作,硬件错误所造成的影响被限制在出错的单元中。我们假设出错后,硬件随即停止工作而不产生错误结果。为了限制错误所产生的影响,硬件被分成错误围堵单元。一旦错误发生,硬件关闭出错的错误围堵单元,同时机器的其它部分对自己进行恢复。一个错误围堵单元必须能够自给自足,所以,它不能小于一个节点,因为节点控制器崩溃会使得整个节点失效。

为了提供硬件缺省围堵,系统内建了一套半独立的单元格,单元格由一个或多个单元组成。每个单元格包括一个监控器代码的完整拷贝,并管理自己的内存映射,用来追踪属于该单元格节点的所有机器内存页。系统中单元不允许直接更改其它单元中的数据结构,这对那些单元的生存是很重要的。一个单元格内的错误只会使得使用该单元格内资源的虚拟机器崩溃,其它的虚拟机不会受到影响。我们建立的系统能够在一个系统成员出错的时候只付出很少的开销(如图 2 所示)。

错误发生后,首先硬件先自我恢复,然后触发所有活动节点的中断,通知系统错误发生了。系统判断哪个单元格和内存页受到了错误的影响。这个出错资源列表用来判定哪个虚拟机受到了错误的影响。受到影响的虚拟机被终止以防止产生不正确的结果。这样,系统能够把错误的影响限制在很小的硬件范围内。

3.2 建立有效的资源管理器

我们建立了有效的资源管理器来对系统资源进行管理,用来支持错误围堵内核的实现。性能良好

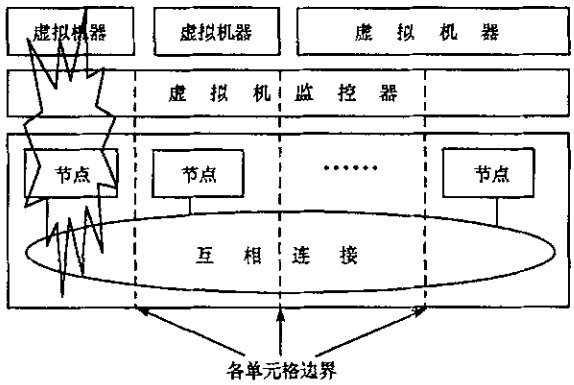

图 2　对单元格内错误进行围堵

Fig.2　Fault containment in cell

的资源管理器能够平衡高效性需求和错误围堵约束之间的关系,提供高性能的同时不影响错误围堵。

一个虚拟机环境中有两个资源管理器,一个是虚拟机监控器,另一个是虚拟机内部运行的操作系统。这两个资源管理器在通常情况下互相不知道对方的情况,自己做自己的决策,所以可能会损害到整体性能。本系统提供的错误围堵和其它系统不同的是:虚拟机监控器很小,这样使出错的概率非常低。虚拟机监控器因此被看作可信任的系统软件层。

系统的资源管理器必须了解错误围堵约束的范围,因为错误围堵可能会和可扩展性相冲突。对于可扩展性来说,资源应该能够在整个系统内广泛应用,避免在特定区域过载,但是,这样会使得虚拟机易受攻击;相反,错误围堵默认的是错误出现的可能性和使用的资源数成正比,因此一个任务所使用的资源相互之间定位应该近一些。对资源管理机制来说,则需要平衡这两个相互冲突的要求。我们通过为每个虚拟机分配有限的单元来平衡这两个互相冲突的需求,提供高性能的同时不影响错误围堵。

3.3　错误围堵的开销

错误容忍技术需要复制资源,这会增加系统开销,而且它不能阻止操作系统崩溃。另一个方法是设置检查点,它不需要复制资源的开销,但是恢复的时间会大大增加,并且一个错误就会导致系统死机,并强制每个应用从最后一个检查点进行恢复。

和以上两种方法相比,错误围堵的开销较少。它把错误的影响限制在系统的一小部分内,系统的其它部分依然能够正常运行,虽然可能会带来系统性能的降级。通过理论计算出来的开销几乎可以忽略不计。

如在文献[10]中指出的,系统引入硬件错误围堵不会带来性能上的开销。我们通过硬件试验长时间连续的工作量来证明该声明。为了测量在正常操作下的开销,我们在虚拟机监控器上使用两种配置:第一种,把系统配置成一个单元,跨越所有 32 个进程;第二种,使用 8 个单元的配置,每个单元内有 4 个进程。两种配置下,系统的工作量是相同的。

实验结果显示,在一个单元的系统中运行的时间和在多个单元系统中的是一样的。这表明如果去掉虚拟所需的开销,系统可以有效地提供错误围堵而不以损害系统性能为代价。

4　实验及结果

我们建立测试平台对系统性能进行测试。为了获得正确有力的实验数据,给系统加上大负荷的工作量,并嵌入非正常的系统调用和系统攻击。

开始时,建立一个应用程序并加入到内核中,它初始化一个特殊的系统调用,该系统调用创建一个进程。系统本身作为一个多线程的内核,它把各线程分散到每个 CPU。各线程与各指派的处理器绑定,防止其它调度影响虚拟机监控器对 CPU 的控制,然后系统进程从操作系统接管控制。保存好操作系统的内核寄存器后,监控器安装它自己的异常处理并接管余下的系统内存。这时操作系统本身处于休眠态,但是如果要它提供设备驱动,它可以随时被唤醒。

系统对每一个任务是这样进行处理的,以 I/O 操作为例,只要有一个虚拟机器请求 I/O 操作,请求会按图 3 所示的过程处理。I/O 请求在系统中产生 trip(1),通过访问控制后,请求被传送给宿主 IRIX(2),存储内核寄存器和异常向量后,要求宿主内核处理对应的 I/O 请求(3)。从宿主操作系统来看,虚拟机监控器进程就像其它正常的内核进程一样,似乎一直在运行。对 I/O 请求的初始化完成后,控制交还给虚拟机监控器,它把宿主机内核恢复成睡眠态。于是硬件产生一个中断(4),该中断由虚拟机监控器来处理,因为这时异常向量已经被改写了。为了让宿主驱动对 I/O 请求进行合适的处理,监控器唤醒处于睡眠态的 IRIX,使它看起来像 I/O 请求才被发送(5)。最后,虚拟机监控器向虚拟机器发送一个虚拟中断,通知它 I/O 请求完成(6)。

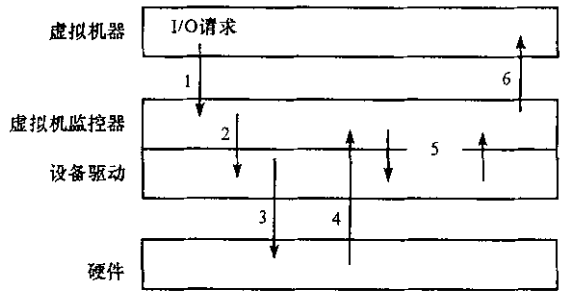

图 3 一个 I/O 请求操作过程

Fig.3 Progress for an I/O request

我们把 32 个进程的工作量加到系统中。完成该工作量,假设计算出来的时间为 1,在不同的虚拟环境下,得到多个测试结果。因为系统被分成多个有虚拟机的独立单元,所以,出错的单元所产生的影响被限制在一定的范围内。实验结果如表 1。

表 1 实验测试结果

Tab.1 Tests results

工作量：32 个进程	计算时间	正常情况下测试时间	攻击成功后测试时间
1 个虚拟机器	1	1	系统崩溃
4 个虚拟机器	1	1.01	1.51
8 个虚拟机器	1	1.04	2.07

5 总 结

本文提出在 IBM 虚拟机器的架构上,使用错误围堵策略建立网络安全设备内核的方法。利用软件和硬件错误围堵技术,防止一个错误引起整个系统的崩溃。系统为资源管理加上一些约束条件,它必须能够调度机器资源,把物理资源返还给虚拟机器,正确处理共享信息,制约对虚拟机器的错误攻击。

我们用以上策略建立了内核模型。给系统加上大负荷,正常和异常的工作。实验结果显示:即使在系统中某些部分出错的情况下,依然不影响系统的整体性能;并且,系统提供错误围堵的同时,不会使系统性能下降,错误围堵的开销几乎可以忽略不计。

参 考 文 献:

[1] Ammann P, Jajodia S, McCollum C D, et al. Surviving Information Warfare Attacks on Databases[A]. Proc. IEEE Symposium on Research in Security and Privacy[C], Oakland CA, May, 1997: 164 – 174.

[2] Ammann P, Jajodia S, Liu P. Recovery from Malicious Transactions[J]. IEEE Transactions on Knowledge and Data Engineering, 2002, 15(2): 1167 – 1185.

[3] Anderson D G, Balakrishnan H, Kaashoek M F, et al. Resilient Overlay Networks[A]. Proc. 18th ACM Symposium on Operating Systems Principles[C], 2001.

[4] Castro M, Liskov B. Practical Byzantine Fault Tolerance[A]. Proc. OSDI[C], 1999.

[5] Yves Deswarte, Laurent Blain, Jean-Charles Fabre, Intrusion Tolerance in Distributed Computing Systems[A]. Proceedings of the IEEE Symposium on Research in Security and Privacy[C], Oakland CA, May, 1991: 110 – 121.

[6] Liu P. Architectures for Intrusion Tolerant Database Systems[A]. Proc. 2002 Annual Computer Security Applications Conference[C], Dec. 2002: 311 – 320.

[7] Jajodia S, McCollum C D, Liu P. Intrusion Confinement by Isolation in Information Systems[J]. Journal of Computer Security, 2000, 8(4): 243 – 279.

[8] Karger P A, Zurko M E, Bonin D W, et al. A Retrospective on the VAX VMM security kerni[J]. IEEE Transactions on software Engineering, 1991, 17(11): 1147 – 1165.

[9] VM Ware Inc. Vmware Virtual Platform[CP]. Available: http://www.vmware.com/products/virtualplatform.html, May 2000.

[10] Teodosiu D. End-to-end Fault Containment in Scalable Shared-memory Multiprocessors[D]. Ph.D. Thesis, Stanford University, 2000.

[11] Teodosiu D, Baxter J, Govil K, et al. Hardware Fault Containment in Scalable Shared-memory Multiprocessors[A]. Proceedings of 24th Annual International Symposium on Computer Architecture (ISCA)[C], 1997, 25: 73 – 84.