基于VMM的Rootkit及其检测技术研究

周天阳 朱俊虎 王清贤
(解放军信息工程大学信息工程学院 郑州 450002)

摘 要 借助虚拟化技术,Rootkit隐藏能力得到极大提升,基于VMM的Rootkit的研究成为主机安全领域的热点。总结了传统Rootkit的隐藏方法和技术瓶颈,介绍了VMM的自身优势和软、硬件实现方法,分析了不同VMM Rootkit的设计原理和运行机制。针对VMM存在性检测的不足,阐述了一种新的VMM恶意性检测思路,同时讨论了VMM Rootkit的演化方向,并从防护的角度提出了一些安全使用虚拟化技术的建议。

关键词 Rootkit,虚拟机监控器,检测,防护
中图法分类号 TP309.5 **文献标识码** A

Research on VMM-based Rootkit and its Detection Technology

ZHOU Tian-yang ZHU Jun-hu WANG Qing-xian
(Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002, China)

Abstract Leveraging virtualization technology, rootkit has improved its stealth capability greatly. Research on VMM-based rootkit has become the focus in computer security field. This paper summarized the traditional hidden methods and the bottleneck of the in-box technology, introduced the advantage of VMM at architecture and the implementation based on software and hardware, and then analyzed the design and operation mechanisms of various VMM Rootkits. In order to resolve the limitation of VMM existence detection, it proposed a new method detecting malicious VMM. In addition, this paper discussed the evolvement of VMM Rootkit, and presented how to apply virtualization techniques safely to defend VMM Rootkit.

Keywords Rootkit, VMM, Detection, Defence

Rootkit技术可以帮助恶意代码隐藏程序属性和攻击行为,躲避反病毒软件的监控,Rootkit及其检测技术是业界关注的焦点。Rootkit一般通过挂钩程序执行路径(Execution Path Hook)[1]、直接内核对象操纵(Direct Kernel Object Manipulation,DKOM)[2]技术篡改操作系统信息,消除入侵痕迹。Rootkit检测根据被修改的系统对象或行为,判别Rootkit的存在。在操作系统内部,Rootkit技术受体系结构束缚,难以向深层发展。

系统虚拟化技术改变了原有计算机的体系结构,虚拟机监控器(Virtual Machine Monitor,VMM)运行于主机硬件之上、操作系统之下,拥有最高特权级,操作系统无法感知自身真实的运行环境。将Rootkit嵌于VMM层,可以利用底层的优势实现深度隐藏。随着虚拟化技术的快速发展,研究人员针对不同虚拟化技术分别提出了3种VMM Rootkit原型——Subvirt[3]、BluePill[4]和Vitriol[5]。这些虽然只是概念验证原型,不具备任何功能性,却在业界引起极大关注。

近年来已有VMM检测技术研究,但同样存在相应的对抗方法。随着虚拟化技术的快速发展,虚拟化产品广泛普及,主机面临新型安全威胁,急需对VMM Rootkit进行研究和分析,提出新型的检测思路和防护方法。

1 Rootkit概述

Rootkit技术使恶意代码隐藏得更深,更容易躲避安全检测。传统Rootkit种类繁多,技术复杂。从用户应用层深入到操作系统内核层,采用的技术主要有以下两种:

一是挂钩程序执行路径(Hook),其主要思想是修改程序执行逻辑,在调用路径的不同层次上,挂钩原有系统函数或指令代码,将其替换为Rootkit自有函数或恶意代码,过滤系统返回信息,为程序执行者提供错误或虚假的结果。

二是直接内核对象操纵(Direct Kernel Object Manipulation,DKOM)。内核对象为用户提供了进程、驱动和网络端口等详细系统信息,DKOM修改了这些对象的关键数据结构,以隐藏与攻击相关的对象信息,提升线程的执行权限。

传统Rootkit技术破坏了操作系统的完整性,引起了系统信息改变和行为异常。通过检测这些系统变化可以判断系统中Rootkit是否存在。继基于特征码的检测技术之后,基于启发式算法的行为识别[6]、系统视图交叉对比[7]、完整性校检[8]等新型操作系统内核检测技术对当前各种复杂的Root-

到稿日期:2011-01-05 返修日期:2011-03-21 本文受国家高技术研究发展计划(863计划)基金资助项目(2008AA10Z419)和河南省基础与前沿技术研究计划(082300410150)资助。

周天阳(1979—),男,硕士生,主要研究方向为网络与信息研究、虚拟化及安全应用,E-mail:zhoutianyang2010@gmail.com;朱俊虎(1974—),男,副教授,主要研究方向为网络与信息安全;王清贤(1960—),教授,博士生导师,主要研究方向为信息安全、算法分析等。

kit 均有较好的检测效果。

为获得系统控制权，攻击和防御都向系统底层迁移，Rootkit 若能够驻留在更底层，就能够躲避甚至控制安全软件。反之安全软件若占据底层，就能够检测、隔离和消除较高层的 Rootkit。传统 Rootkit 面临两个主要的问题：一是不能完全控制整个系统，因为 Rootkit 与检测机制共处于操作系统最底层，同时拥有最高特权级的绝对优势；二是无法平衡功能性和隐藏性的取舍，功能越是强大的 Rootkit 调用的系统资源越多，越容易被检测到。

2 基于 VMM 的 Rootkit

虚拟化技术是计算机系统的核心技术，具体表现为对系统逐层抽象，下层为上层提供接口并拥有对上层的控制权。借助系统虚拟化技术，基于 VMM 构建的 Rootkit 可以独立于操作系统之外运行，其突破了操作系统的限制，同时获得了更高的特权级，从而实现深度隐藏。根据 VMM 不同的实现方式，Rootkit 可基于软件虚拟化技术构造自身，也可利用硬件辅助虚拟化技术进行隐藏。

2.1 虚拟机监控器(VMM)

VMM 是在现有系统中插入的一薄层虚拟化监控软件，其截获上层操作系统的特权、敏感指令和中断、异常等系统事件并模拟处理，从而实现对硬件资源的控制、抽象、封装和隔离，使操作系统如同直接运行在硬件平台上。VMM 还可以在多个客户机操作系统环境之间划分同一套硬件资源，提供对计算平台的复制，如图 1 所示。

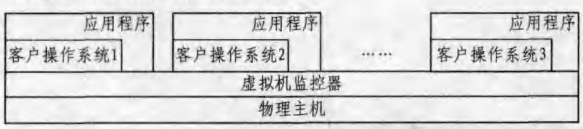


图 1 含有 VMM 的计算机系统层次结构

最初，VMM 以纯软件模拟方式构建。VMM 寄生在某个宿主操作系统中，利用宿主提供的设备驱动及底层服务，这种实现方式称为宿主模型(Host-based Model)[9]。宿主模型的 VMM 采用特权级压缩技术，以保证自身拥有系统最高特权级，即 VMM 运行在 Ring 0 级，客户操作系统运行在 Ring 1 或 Ring 3 级。由于 x86 指令体系在设计之初没有考虑到虚拟化，某些敏感指令可直接在非特权级执行而不会触发异常，这样 VMM 就不能截获这些指令，导致了所谓的“虚拟化漏洞”问题。二进制代码翻译技术可以解决这个问题，但会带来较大的系统开销。

为完全高效地控制系统，VMM 需要直接运行在底层硬件上，并拥有最高特权级，这种不借助宿主操作系统的 VMM 称为监控器模型(Hypervisor Model)。近年来，硬件厂商在各自的产品中提供了虚拟化支持，以 Intel VT-x[10] 为例来说明硬件辅助虚拟化的技术原理。CPU 引入了一种称为 VMX (Virtual Machine Extensions)的新操作模式和一套虚拟化指令集。CPU 执行 VMXON 指令后进入 VMX 模式，执行 VMXOFF 指令后退出该模式。CPU 在 VMX 模式中可在根(root)与非根(non-root)两种工作状态间切换，如图 2 所示。VMM 运行在根态，而客户操作系统运行在非根态。客户机执行敏感指令引发 VM 退出(VM-Exit)，CPU 自动从非根态切换到根态。VMM 模拟指令执行后，将控制权返回给客户机，CPU 进入到非根态，这称为 VM 进入(VM-Entry)。为控制客户机运行，VT-x 还引入了虚拟机控制结构(Virtual-Machine Control Structure，VMCS)。VMCS 详细记录了 VM 退出条件、退出原因、退出时客户机的状态和需装入的宿主机 VMM 的状态等控制信息。

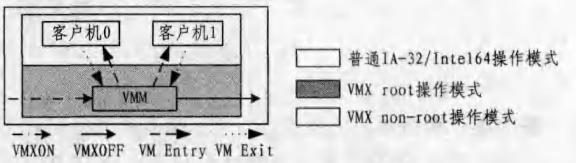


图 2 Intel VT-x VMX 操作模式

在两种操作状态下，CPU 都有相应的 Ring 0～Ring 3 的特权级划分。这样 VMM 不仅获得了系统最高特权级，还保证了客户操作系统可以运行在 Ring 0 级，避免了特权级压缩带来的问题。借助于硬件辅助虚拟化技术，VMM 可以直接而高效地控制底层硬件。

2.2 基于虚拟机的 Rootkit(Virtual Machine based Rootkit，VMBR)

VMBR 是以软件虚拟化方式实现的 VMM Rootkit，采用宿主模型。VMBR 通过在现有操作系统下安装一个 VMM (如 Virtual PC[3])，然后将恶意服务构建在一个隔离的虚拟机中，使目标主机很难检测到。

VMBR 需要更改操作系统的启动顺序，将自身插入到目标操作系统的底层。在主机启动时，VMBR 首先获取系统权限加载自身，之后正常启动目标操作系统，并将其作为客户操作系统运行在自己之上，这样 VMBR 就运行在目标操作系统与物理硬件之间。VMBR 成功加载后，利用自己的底层优势，借助虚拟机监控器，实现对目标操作系统的监视和完全控制，而目标操作系统很难感知 VMBR 的存在。VMBR 的整体结构如图 3 所示，阴影部分为各组件。

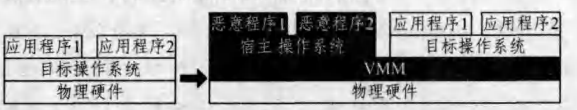


图 3 感染 VMBR 后目标系统运行模式转换图[3]

攻击者可以通过多种手段获取足够的系统权限，以修改系统的启动顺序，如远程漏洞利用或邮件欺骗等。VMBR 还需要永久保存在目标主机的磁盘上，如果目标操作系统是 Windows XP，则安装在第一个活动分区的开始；如果是 Linux 操作系统，则可以禁用交换分区，然后利用交换分区来存储[3]。然后，VMBR 修改系统的主引导分区的启动顺序，确保先于目标操作系统加载。

与传统 Rootkit 相比，VMBR 在平衡隐藏性和功能性方面具有优势，一旦安装成功，可以运行多种恶意程序。这是因为 VMBR 集成了一个经过修改的操作系统内核作为宿主，利用这个独立的宿主操作系统，可以构建任何恶意服务模块，而这一切对目标操作系统是不可见的。

2.3 基于硬件辅助虚拟化技术的 Rootkit(Hardware-assist Virtualization based Rootkit，HVBR)

HVBR 利用 CPU 的虚拟化扩展功能，将直接运行在硬件上的目标操作系统动态地迁移到 Hypervisor 上，由 Hypervisor 获得对操作系统的完全控制。HVBR 一般基于精简监控器(Thin Hypervisor)模型[4]，与 VMBR 的 VMM 不同，

Thin Hypervisor 并不虚拟所有的底层硬件，目标系统甚至可以直接访问内存和 I/O 设备。HVBR 根据需要，有选择地构造一些敏感指令，如 CPUID，INVD，MOV CR3 操作等。这些敏感指令将引起客户 VM 退出，HVBR 利用 Hypervisor 对退出事件的处理实现对上层的监控。

HVBR 以驱动的方式动态载入运行时的目标操作系统。安装时，首先检查 CPU 是否支持虚拟化。如果支持，则设置相关寄存器控制位，开启虚拟化功能，进入 VMX 根操作模式，为 VMCS 分配内存空间并初始化，将程序计数器(PC)置为操作系统调用 HVBR 代码之后的下一条指令，为以后 VM 退出事件准备 Hypervisor 处理程序，并切换到 VMX 非根操作模式，恢复目标操作系统正常运行，继续执行 PC 中的下一条指令。此时，目标操作系统已被迁移至虚拟机中，完全受 HVBR 控制。HVBR 的程序流程如图 4 所示。

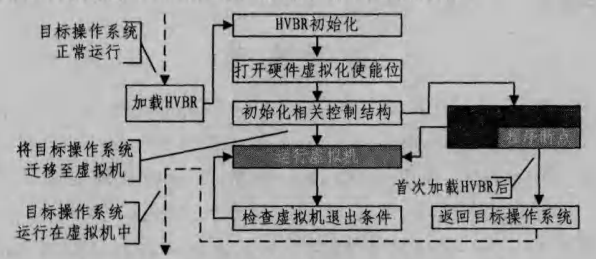


图 4　HVBR 执行流程[4]

HVBR 无需修改 BIOS、启动扇区和系统文件，不需要重启操作系统。由于很少干涉目标系统的操作，系统性能也不会受到明显影响，因此获得了比 VMBR 更好的隐藏性。但在系统重启后会失效，且无法实现永久驻留。

2.4　VMM Rootkit 小结

VMM Rootkit 以获取系统最高特权级为目标，插入到目标操作系统底层进行隐藏。VMM Rootkit 可以构建的恶意服务包括构造特殊的指令序列篡改当前进程，插入中断或异常改变程序执行流程，过滤、监视、记录对内存和 I/O 设备的访问，建立隐蔽通信等。目前，已知的 VMM Rootkits 均是概念验证性模型，并不包含恶意功能，实现技术各有不同，如表 1 所列。

表 1　已发布的 VMM Rootkits 之间的差异

差异类别	Subvirt	Vitriol	BluePill	NewBluePill
虚拟化平台	Virtual PC/VMware	Intel VT-x	AMD SVM	Intel VT-x 和 AMD SVM
虚拟化种类	软件虚拟化	硬件辅助虚拟化	硬件辅助虚拟化	硬件辅助虚拟化
VMM 模型	混合模型	瘦监控器模型	瘦监控模型	瘦监控模型
目标操作系统	Windows XP 或 Linux	MacOS X	Windows Vista X64	WindowsVista 和 Linux
运算环境	32 bit	64 bit	64 bit	64 bit
装载方法	重启操作系统，改变系统引导顺序，先于操作系统加载	在系统运行时，以可装载模块的方式加载	在系统运行时，以驱动方式加载	在系统运行时，以驱动方式加载
存活性	驻留硬盘引导分区，可长期驻留	系统运行期间存活，重启后无效	系统运行期间存活，重启后无效	系统运行期间存活，重启后无效
技术特点	性能开销较大，影响用户操作	性能开销小，对用户透明不支持嵌套虚拟化	性能开销小，对用户透明支持嵌套虚拟化	性能开销小，对用户透明支持嵌套虚拟化

3　VMM Rootkit 的检测

VMM Rootkit 在体系结构上具备底层优势，但事实上不存在一个理想化的 VMM 对操作系统彻底透明，VMM Rootkit 的运行仍然会给系统带来某种变化和负效应，可以据此检测 VMM Rootkit。

3.1　虚拟化使能位读取检测

HVBR 在安装时必须先开启系统虚拟化功能，即将 CPU 控制寄存器(IntelVT-x：CR4. VMXE，AMD SVM：EFER_MSR. SVME)虚拟化使能标志置位。检测时，执行 RDMSR 指令读取寄存器，如果该标志位返回结果为 1，则证明 VMM 的存在。然而，RDMSR 属于特权指令，可能被 VMM Rootkit 截获和欺骗，导致该方法失效。

3.2　计时检测

VMM Rootkit 运行时，会截获和模拟执行一些敏感指令(如读取 CR3)，引起 CPU 操作模式的转换。CPU 需要花费大量时钟周期处理 VM Entry 和 VM Exit 事件，从而导致指令执行时间延迟。所以在目标操作系统中构造执行特定的指令序列，计算实际耗时，可以检测 VMM 是否存在[11]。为获取精确的计时结果，检测程序经常需要执行大量敏感计时指令。然而，频繁的计时操作会引起 VMM Rootkit 的警觉。为躲避检测，VMM Rootkit 先将自己卸载并休眠，过一定时间后，VMM Rootkit 再被唤醒并重新加载[12]。

3.3　计数检测

计数检测由计时检测发展而来。计数检测基于双核的 CPU，通过随机线程指派，一个核用来执行独立计数线程，一个核执行测试线程，其中包含能够导致 VMEXIT 事件发生的敏感指令。当计数核上的计数超过某阈值时，就认定 VMM 的存在[13]。如果 VMM Rootkit 可以设法通过核间中断，干扰独立内核的计数，则这种检测方法可能失效。

3.4　基于 CPU 内部资源视图差异的检测

这种方法利用 CPU 的缓存机制进行检测。VMM 截获敏感指令会引起 CPU 模式的切换，此时缓存中的记录会被修改或刷新，对相同地址进行再次访问的结果将发生改变，这就是 VMM 带来的缓存一致性问题。由于无法直接读取缓存记录，一般通过间接的方式来判断缓存记录是否被修改。如检测程序在执行敏感指令前，可以修改与 TLB 对应的页表项，使其指向新的物理页。如果存在 VMM，再次访问相同线性地址空间时，由于 TLB 被刷新，需要重查页表，则访存操作返回的是新的物理页内容；否则，TLB 被命中，返回的仍然是旧页面内容[14]，这是一种基于 TLB 视图差异的检测方法。相关的检测方法还有基于 TLB 程序执行时间差异的检测[14]、Cache 失效指令检测[15]和 RSB 记录刷新检测等[16]。

然而，不同型号的 CPU 缓存的设计和实现各有不同，缓存部件组织结构复杂，不存在通用的检测方法适用不同型号的 CPU，且检测程序自身运行时也会导致缓存的刷新，影响检测效果。

上述检测方法都试图通过在一个原本非虚拟化的系统中判断 VMM 的存在来检测 VMM Rootkit。随着 VMM 软、硬件产品的普及，即使检测出了 VMM，也不能说明 VMM Rootkit 的存在，还需要判断 VMM 的恶意性。

3.5 VMM 恶意性检测技术

VMM 恶意性检测是发现 VMM Rootkit 最直接、最有效的方式，一般采用内存扫描检测。具体需要解决两个问题：一是能够访问到 VMM 所在的地址空间；二是分析 VMM 是否具有恶意性。

VMM Rootkit 一般使用自己的页表，且与目标操作系统的线性地址空间隔离。为获取 VMM 代码，只能直接访问物理内存。可以采用页表(PTE)补丁技术，修改页表项的指针，使其指向特定物理内存页。在获得 VMM 物理内存的映像后，需要进一步分析 VMM 的恶意性。本文认为枚举 VMM 控制结构(如 VMCB 和 VMCS)的个数可以方便、快捷地进行判断。因为，VMM Rootkit 不可能寄生在原有 VMM 中实现恶意功能，只会采用虚拟化嵌套的方式重建一个 VMM 供自己使用。如果检测出系统中存在多于预期的 VMM，则说明这必定是 VMM Rootkit。VMM 控制结构的版本标识和特定数据域可作为检测依据，本文选取了 VMCS 中的 Reivsion_identifier 和 Geust_CR0 作为联合特征进行全物理内存的搜索。检测结果表明，本文方法能够快速地检测出 VMM Rootkit 的存在。

VMM Rootkit 为抵御基于 PTE 补丁的物理内存扫描，会使用影子页表(Shadow Page Table)或硬件支持的嵌套页表(Nested Page Table，NPT)、扩展页表(Extension Page Table，EPT)技术进行对抗。检测程序修改 PTE 的操作容易引起 #PF(页失效)异常。VMM Rootkit 捕获该异常后，可以将指向自身的页表项重定向到垃圾页面。

4 VMM Rootkit 的演化和预防

随着虚拟化技术快速发展，VMM Rootkit 将不断升级。利用 Intel VT-d 和 AMD IOMMU 的 I/O 虚拟化技术，VMM Rootkit 可以进行 DMA 重定向，致使所有 DMA 设备无法直接读写物理内存进行检测。目前，VMM Rootkit 主要以 x86 体系架构为攻击对象，SPARC，POWER 等其它平台也提供虚拟化功能，开展基于这些平台的 VMM Rootkit 及其检测研究是非常必要的。

VMM Rootkit 一旦成功运行，主机的完整性即遭受破坏，检测受到各种对抗手段的干扰变得非常困难。因此，研究如何预防 VMM Rootkit 加载进系统显得更为重要。

通过恢复磁盘启动扇区或者利用光盘、Ukey 等安全介质启动计算机，可以防止 VMBR 的安装。对于 HVBR，利用系统的驱动保护机制可以禁止其加载，如 Windows Vista 和 Win 7 等只允许经过数字签名的驱动程序加载到内核中。但是，HVBR 利用系统缺陷仍然可以获得内核访问权限，绕过驱动保护机制进行安装[4]。

另一种可行的防护措施是通过硬件设置或者软件授权来限定虚拟化的安全使用。如 Intel CPU 的特殊模式寄存器 IA32_FEATURE_CONTROL 中的 LOCK 位若被置 0，则 CR4. VMXE 位无法置 1。因此，设置 BIOS 使 CR4. VMXE=0 且 LOCK=0，可以锁死和禁用 CPU 虚拟化功能[10]，防止 HVBR 的加载。AMD 平台则为用户提供了一个 64 位密钥，用于在 BIOS 被锁死时重启 SVM。

可信平台模块(Trusted Platform Module)也可用来防范 VMM Rootkit。在 Intel 平台上，设置 IA32_FEATURE_CONTROL 的相应位可以限定 VMXON 指令只能在 SMX(Safer Mode Extensions)操作模式中执行，否则就会触发通用保护异常 #GF[10]。执行 GETSEC[SENTER]指令进入 SMX，在 TPM 的认证保护下可以安全使用硬件虚拟化功能。

结束语 本文总结了 VMM Rootkit 的设计思路和技术特点，从 VMM 存在性检测和 VMM 恶意性检测两个角度研究了 VMM Rootkit 的检测技术及其对抗手段，提出了一种基于物理内存扫描的新思路。同时，展望 VMM Rootkit 可能的发展方向，并给出防御 VMM Rootkit 的方法。

虚拟化技术正广泛应用于主机安全领域，VMM 可用于构建入侵检测系统、访问控制系统、蜜罐和调试器等。深入研究 VMM Rootkit 相关技术，将为 VMM 的安全应用提供重要参考。

参 考 文 献

[1] Rutkowska J. Rootkits vs Stealth by Design Malware [C/OL]. http://invisiblethings.org/papers/rutkowska-bheurope2006.ppt

[2] Hoglund G, Butler J. Rootkits——Windows 内核的安全防护[M]. 韩智文，译. 北京：清华大学出版社，2007：163-263

[3] King S T, Chen P M, Wang Y, et al. Subvirt: Implementing malware with virtual machines [C]// Proceeding of the 2006 IEEE Symposium on Seurity and Privacy. Berkeley/Oakland, California, IEEE CS digital library, 2006: 314-327

[4] Rutkowska J, Tereshkin A. Subverting Vista Kernel For fun and Profit [C/OL]. Black Hat USA. http://blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf, 2006-08

[5] Zovi D Z. Hardware Virtualization Rootkits. Black Hat USA [C/OL]. http://blackhat.com/ presentations/ bh-usa-06/BH-US-06-Zovi.pdf

[6] 龙海，郝东白，黄皓. 系统服务 Rootkit 隐藏行为分析[J]. 计算机科学，2008，35(6)：103-106

[7] 白光冬，郭耀，陈向群. 一种基于交叉视图的 Windows Rootkit 检测方法[J]. 计算机科学，2009，36(8)：133-137

[8] Rutkowska J. System Virginity Verifier-Defining the Roadmap for Malware Detection on Windows System, Hack In The Box [C/OL]. http://invisiblethings.org/papers/hitb05_virginity_verifier.ppt, 2005

[9] Goldberg R P. Survey of Virtual Machine Research[J]. IEEE Computer, 1974, June: 34-45

[10] Intel. Intel® 64 and IA-32 Architecture Software Developer's Manual; Volume3B: System Programming Guide, Part2, 2007. [EB/OL]. http:// download.intel.com/design/processor/manuals/253669.pdf

[11] Garfinkel T, Adams T. Compatibility is Not Transparency: VMM Detection Myths and Realities[C]// Proceeding of the 11th USENIX Workshop on Hot Topics in Operating Systems. CA, USA: USENIX Association Berkeley, 2007: 1～6

[12] Rutkowska J. Is Game Over(), Anyone?. Black Hat USA [C/OL]. https://www.blackhat.com/ presentations/bh-usa-07/Rutkowska/Presentation/bh-usa-07-rutkowska.pdf, 2007

[13] Ning J, Wang H, Guo Shize, et al. CBD: A Counter-based Detection Method for VMM in Hardware Virtualization Technology [C]//2010 First International Conference on Pervasive Computing, Signal Processing and Applications. Harbin, China: IEEE

CS,2010:352-356

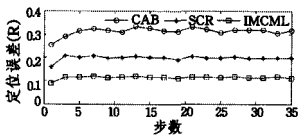
[14] Adams K. BluePill detection in two easy steps [C/OL]. http://x86vmm. blogspot. com/2007/07bulepill-detection-in-two-easy-steps. html

[15] Ptacek T, Lawson N, Ferrie P. Don't Tell Joanna, The Virtualized Rootkit Is Dead. Black Hat USA [C/OL]. https://www.blackhat. com/presentations/bh-usa-07/Ptacek _ Goldsmith _ and _ Lawson/Presentation/bh-usa-07-ptacek _ goldsmith _ and _ lawson. pdf,2007

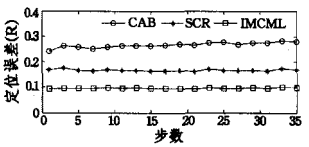
[16] Bulygin Y. CPU side-channels vs. virtualization rootkits: the good, the bad, or the ugly. ToorCon Seattle'08 [C/OL]. http://c7zero. info/stuff/ hyper-channel_toorcon_seattle. ppt,2008

(上接第 64 页)

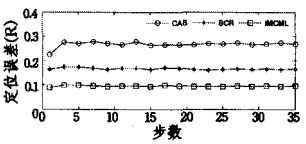
向具有很强的随机性,因此导致定位稳定性及定位精度较差。在随机路点模型中,导标经过仿真中心区域的概率较高,未知节点易于收到更多的定位信息,故定位效果比其他两种移动模型要好。如图 10 所示,SCR 法和 IMCML 法比 CAB 法有更好的定位稳定性,而 IMCML 法的定位精度最好。



(a) 随机路径模型



(b) 随机路点模型



(c) 随机方向模型

图 10 (a)为随机路径模型 vs 定位误差;(b)随机路点模型 vs 定位误差;(c)随机方向模型 vs 定位误差

综上所述,通过在不同条件下对提出的算法进行仿真验证,文中提出的 IMCML 法的定位误差最小,且算法稳定性较好;SCR 法简便易行,定位精度比 CAB 法要好;CAB 法的定位效果和稳定性均不如其他两种算法。

结束语 本文将多能量级的思想引入到动态网络,提出了一种基于多能量级的改进 Monte Carlo 的移动节点定位算法,并分析了多能量级节点定位算法在动态网络中的有效性及限定区域随机投点法的实用性。通过改变不同的仿真条件,对 3 种节点定位方法的定位精度及算法稳定性进行了分析比较,算法均表现出了良好的定位效果。另外,提出了一种导标共线度约束策略,引入了共线度限制因子(CLF),很好地解决了导标共线问题。仿真结果表明,通过改变 CLF 值,提出的移动节点定位算法的定位精度有了显著提高,算法在低导标密度及高移动速度下均具有较高的定位精度,具有较好的适应性和稳定性。

参 考 文 献

[1] Akyildiz I F, Su W, Sankarasubramaniam Y, et al. Wireless sensor networks: a survey[J]. Computer Networks, 2002, 38(4): 393-422

[2] Frankie K W, So H C. Accurate distributed range-based positioning algorithm for wireless sensor networks[J]. IEEE Transaction on Signal Processing, 2009, 57(10): 4100-4105

[3] Wang Y, Wang X D, Wang D M, et al. Range-free localization using expected hop progress in wireless sensor networks [J]. IEEE Transaction on Parallel and Distributed Systems, 2009, 20(10): 1540-1552

[4] Hu L X, Evans D. Localization for mobile sensor networks[C]// Proc. of the 10th Annual Int'l Conf. on Mobile Computing and Networking. Philadelphia, PA: USA, 2004: 45-57

[5] Zhang S G, Cao J N, Chen L J, et al. Accurate and energy-efficient range-free localization for mobile sensor networks [J]. IEEE Trans. on Mobile Computing, 2010, 9(8): 897-910

[6] 彭鑫,李仁发,罗娟. 一种基于非度量多维标度的移动定位算法[J]. 计算机科学,2008,35(10):219-222

[7] Wang W D, Zhu Q X. Sequential Monte carlo localizartion in mobile sensor networks [J]. Wireless Networs, 2009, 15(4): 481-495

[8] Sheu J P, Hu W K, Lin J C. Distributed localization scheme for mobile sensor networks[J]. IEEE Transaction on Mobile Computing, 2010, 9(4): 516-525

[9] 石琴琴,霍宏,方涛,等. 使用最速下降算法提高极大似然估计算法的节点定位精度[J]. 计算机应用研究,2008,25(7):2038-2040

[10] Vivekanandan V, Wong V W S. Concentric anchor beacon localization algorithm for wireless sensor networks[J]. IEEE Transaction on Vehicular Technollogy, 2007, 56(5): 2733-2744

[11] Liu K Z, Cui Y Q, Zhang J F, et al. Anchor selection scheme for multi-energy level localization method in wireless sensor networks[C]// Proceedings of 4th International Conference on Ubiquitous Information Technologies & Applications. Fukuoka: Japan, 2009: 297-300

[12] Fidan B, Drake S P, Anderson B D O, et al. Collinearity problems in passive target localization using direction finding sensors[C]// Proceedings of 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing. Melbourne: Australia, 2009: 115-120

[13] Bettstetter C, Resta G, Santi P. The node distribution of the random waypoint mobility model for wireless ad hoc networks[J]. IEEE Transaction on Mobile Computer, 2003, 2(3): 257-269

[14] 刘宴涛,安建平,卢继华,等. 无线自组网个体移动模型分析[J]. 通信学报,2010,31(2):36-43

[15] Carofiglio G, Chiasserini C F, Garetto M, et al. Route stability in MANETs under the rand direction mobility model[J]. IEEE Transaction on Mobile Computing, 2009, 8(9): 1167-1179