硬件虚拟化 rootkit 检测方法研究综述

施江勇[1], 王会梅[1], 鲜 明[1], 荣 宏[1], 金从军[2]
(1. 国防科学技术大学 电子信息系统复杂电磁环境效应国家重点实验室, 长沙 410073; 2. 航天系统仿真实验室, 北京 100854)

摘 要: 随着虚拟化技术的发展及其在云计算中的广泛应用,传统的 rootkit 也开始利用硬件虚拟化技术来隐藏自己。为了对抗这一新型 rootkit 的攻击,研究了传统 rootkit 检测方法在检测硬件虚拟化 rootkit(HVMR)上的不足,分析了现有的 HVMR 检测方法,包括基于指令执行时间差异的检测方法、基于内存资源视图差异的检测方法、基于 CPU 异常和错误的检测方法,以及基于指令计数的监测方法等。总结了这些检测方法的优缺点,并在此基础上提出了两种通过扫描内存代码来检测 HVMR 恶意性的方法,分别是基于 hypervisor 的恶意性检测方法和基于硬件的恶意性检测方法,同时也预测了未来虚拟化检测技术的发展方向。

关键词: 硬件虚拟化; rootkit 检测; 基于硬件虚拟化的 rootkit; Bluepill 检测; 恶意性检测; 内存扫描
中图分类号: TP393.08 **文献标志码:** A **文章编号:** 1001-3695(2014)01-0001-05
doi:10.3969/j.issn.1001-3695.2014.01.001

Summarize of detection methods on hardware-based virtualization machine rootkit

SHI Jiang-yong[1], WANG Hui-mei[1], XIAN Ming[1], RONG Hong[1], JIN Cong-jun[2]
(1. *State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics & Information System, National University of Defense Technology, Changsha* 410073, *China*; 2. *Science & Technology on Space System Simulation Laoratory, Beijing Simulation Center, Beijing* 100854, *China*)

Abstract: The development of virtualization technology and its widely application in cloud computing promotes the traditional rootkit making use of hardware virtualization to hide itself. To confront this new threat, this paper studied the deficiency of traditional rootkit detection ways in detecting hardware-based virtualization machine rootkit (HVMR), analyzed the merits and demerits of existing methods in detecting HVMR, and concluded in incapable of judging the malicious of HVMR. Based on this, it presented two methods to detect the malicious of HVMR, namely hypervisor-based malicious detection and hardware-based malicious detection. Finally, it predicted the future developments of virtualization detection technology.

Key words: hardware-based virtualization; rootkit detection; HVMR; Bluepill detection; malicious detection; memory scan

1 研究背景

虚拟化技术作为云计算的基础技术,近年来伴随着云计算的发展得到了广泛和深入的研究。虚拟化环境(VME)的安全和对抗问题也越来越得到企业和研究人员的重视,其中基于虚拟化的 rootkit(VMBR)的出现使得传统的系统安全面临新的挑战。VMBR 利用虚拟化技术将用户的合法操作系统虚拟化为其上的一个虚拟机,而恶意代码运行于虚拟机监控器层(hypervisor),隐蔽性很高,难于检测,其典型代表是 SubVirt[1]。硬件虚拟化技术的出现大大提高了虚拟化的性能,但同时也出现了基于硬件虚拟化技术的 rootkit(HVMR),这种 rootkit 利用硬件(尤其是 CPU)对虚拟化的支持拦截虚拟机的某些指令,处理后再返回到虚拟机中。由于 HVMR 支持虚拟机对 I/O 设备和内存的直接访问,对于系统的干扰较少,其隐蔽性较之 VMBR 更高,危害也更大,其典型代表是 BluePill[2] 和 Vitriol[3]。本文通过研究这一类 rootkit 的检测方法和隐藏机制,分析已有检测方法的有效性,研究 rootkit 的反检测措施,通过分析检测与反检测的博弈过程理清 HVMR 检测技术的发展脉络,预测未来技术的发展方向,为云计算环境下的系统安全问题提供一个解决思路。图 1 为研究对象之间的关系。

1.1 硬件虚拟化技术

硬件虚拟化技术在硬件层面上提供对虚拟化的直接支持,通过这种设计提高虚拟效率、降低开发难度。其基本思想是对硬件本身加入足够的虚拟化功能,如 Intel VT-x 技术为 CPU 增加了专门的虚拟化执行指令,进而拦截操作系统对敏感指令的执行及对敏感资源的访问,再通过异常的方式报告给 VMM。这方面的典型代表还有 Intel VT-d、AMD SVM 和 AMD IOMMU。图 2 为带 VT-d 功能的平台架构图[4]。

硬件虚拟化技术提供了硬件层面的虚拟化功能,通过为 CPU 提供虚拟化执行指令,避免了翻译二进制代码的巨大工作量,提高了虚拟化指令的执行效率,通过在硬件上实现内存地址甚至是 I/O 设备的映射以及支持二次寻址功能,大大简化

收稿日期: 2013-04-18; **修回日期:** 2013-06-04

作者简介: 施江勇(1990-),男,硕士,主要研究方向为网络安全(sjy.paper@gmail.com);王会梅(1981-),女,讲师,博士,主要研究方向为网络安全、电子信息系统建模仿真与评估;鲜明(1970-),男,研究员,博导,主要研究方向为电子信息系统建模仿真与评估、网电空间对抗;荣宏(1988-),男,硕士,主要研究方向为网络安全;金从军(1970-),男,研究员,主要研究方向为射频仿真.

了虚拟机监控器层和虚拟机的实现过程。另一方面,rootkit 技术也可以利用硬件虚拟化技术提高其隐藏深度,使得针对虚拟化 rootkit 的检测更加复杂。

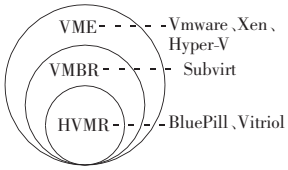

图 1 虚拟化环境(VME)、基于虚拟化的 rootkit(VMBR)、基于硬件虚拟化的 rootkit(HVMR)的关系

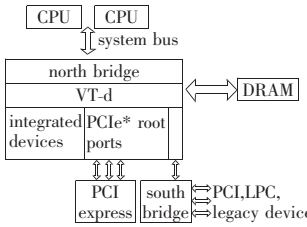

图 2 支持 Intel VT-d 功能的平台架构

1.2 基于硬件虚拟化技术的 rootkit

硬件虚拟化 rootkit(HVMR)首先出现于 2006 年的黑帽大会上,其典型代表包括利用 AMD SVM(secure virtual machine)硬件虚拟化技术的 BluePill[2],和利用 Intel VT(virtualization technology)硬件虚拟化技术的 Vitriol[3],分别由 Rutkowska 和 Zovi 提出。HVMR 紧随硬件虚拟化技术的出现而出现,其诞生甚至在硬件虚拟化技术大规模用于合法的服务器虚拟化之前(如 Xen、vSphere 等),这使得研究人员一开始就将对硬件虚拟化环境的检测等同于对 HVMR 的检测,而随着虚拟化技术,包括硬件辅助虚拟化技术的大量推广应用,仅仅是检测出虚拟化环境并不能判断该环境是否是恶意的。因此,本文在分析现有的检测硬件虚拟环境技术的基础上,提出几种判断虚拟环境是否恶意的方法,并对未来的技术发展做了一个预测和展望。

图 3 为 BluePill 的实现思路[2]。最左侧表示当操作系统启动 BluePill 后,BluePill 启用硬件虚拟化技术,设置相应虚拟机控制块(virtual machine control blocks,VMCB)中的内容,填充其中的 RIP(指令寄存器)项指向物理机下一条需要运行的指令(最右侧),然后通过 VMRUN 指令正式启动虚拟机。当发生指定事件从虚拟机中陷入到 hypervisor 中需要处理时,BluePill 会检查退出原因,并在处理完相应事件后通过 VMRUN 恢复虚拟机的执行,此时 RIP 已被硬件自动修改为上次退出虚拟机状态时最后执行的指令地址,因此保证了返回虚拟机后继续执行。通过在退出虚拟机后的事件处理中增加恶意代码,即可实现窃取数据与控制系统的目的。而且由于恶意代码运行于系统之外,使得对其的检测很困难。

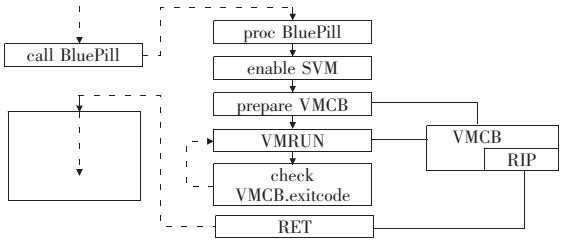

图 3 BluePill 的实现原理

1.3 传统的 rootkit 检测技术的局限性

表 1 总结了传统的 rootkit 技术的分类[5]。不同于传统的通过修改程序执行路径的 rootkit,HVMR 基本上不涉及包括地址表、服务调度表、中断描述符表以及内联函数等与程序执行路径相关的结构的更改,这样就使得 rootkit 的隐蔽性更高,而高隐蔽性正是 rootkit 的最大威胁。HVMR 也不同于直接内核对象操作的 rootkit(DKOM),DKOM 技术通过绕过系统的对象管理器实现对内核对象的更改,从而实现对特定资源的隐藏,其特权级至多是与系统平级,有很多方法可以检测,而 HVMR 则通过将合法的系统虚拟化,使得恶意代码的特权级高于系统内核,因此 HVMR 更具有破坏性。

表 1 传统 rootkit 的分类

类型	特征	代表
应用级 rootkit	修改地址表、服务调度表、中断描述符表以及内联函数等与执行路径相关的结构	Hacker Defender
内核级 rootkit	绕过操作系统的对象管理器直接对内核对象进行修改,实现隐藏进程、设备驱动程序、端口以及提升进程权限等功能	AgonyRing0 FU DKOM

传统的 rootkit 检测,如基于数据完整性的检测,其本身有效执行的前提是检测 rootkit 的代码运行于比 rootkit 高的特权级之上,或者至少与 rootkit 同级的特权级之上,否则其检测就不具有可信性。这是因为高特权级的 rootkit 可以通过 hook 技术挂钩系统文件、进程、注册表、服务等的查询函数,从而有效躲避这种检测方法。即便是针对内核 rootkit 的基于执行路径分析的检测方法,为了调试分析指令的执行,CPU 必须处于单步模式,所以每执行一条指令就触发一次中断,严重影响了系统的性能。HVMR 可以据此判断检测是否正在执行,从而通过欺骗和静默的方法来躲避检测。表 2 总结了传统 rootkit 检测方法的原理和缺点。

表 2 传统的 rootkit 检测方法及其局限性

检测方法	描述	局限性
基于特征码的检测	扫描被检测文件中是否含有特征数据库中的恶意 rootkit 代码特征串	依赖于已有的特征库,只能对已知的 rootkit 起作用
基于地址分析的检测	分析静态表和程序分支中的地址是否超出某一可接受的范围	误判较多,对于系统自身的良性钩子(如防病毒软件、调试软件等)误判较多
基于交叉视图的检测[6]	对比操作系统的不同层次的进程、文件、驱动和端口等信息	只能检测应用层 rootkit,对于内核态 rootkit 和 HVMR 无能为力
基于可执行路径分析的检测	对系统服务运行时所用的指令进行计数,以判断是否有额外的代码执行	对系统性能影响较大,另外指令计数器可能遭到 HVMR 的窜改
基于完整性和签名检验的检测	将当前文件系统或内存的快照与一个已知且可信的基准进行比较	基准快照可能遭到了 rootkit 的窜改,使得检测结果并不准确

2 HVMR 的存在性检测方法

BluePill 的设计标准中提到,即便公开其实现的源代码,也无法对其实现检测。然而几个月之后的 SyScan 会议上 Barbosa[7] 提出了应对 BluePill 的检测方法,并提出 HVMR 不可能完全控制所有的计算机资源,这些资源包括 TLB、转移预测指令、SMP 进程等,利用这些未受控制的资源提出了相应的几种检测方法,并进行了有效性分析。Ptacek 等人[8] 针对 Vitriol 提出了相应的检测方法,并声称这些方法同样适用于检测 BluePill。不久,在 2007 年的黑帽大会上 Rutkowska[9] 针对之前出现的各种检测 BluePill 的方法一一进行了有效性分析,提出其中的很多方法并不实用,并提出了时间欺骗和 Blue Chicken 等反检测措施。之后关于 HVMR 的检测与反检测技术的研究,如文献[10~14]等,基本上是沿着这几个报告的路线走,对其进行补充和实践。可以预测针对 HVMR 的检测与反检测机制的博弈将持续下去,共同推进虚拟化安全的发展。综合各类文献的分析,本文试图理清 HVMR 检测与反检测技术发展的线路,并对未来的技术发展作出预测。

2.1　基于指令执行时间差异的检测方法

基于指令执行时间差异的检测方法通过大量执行指令,测量指令执行消耗时间来判断。这种检测基于虚拟机执行时的以下因素[14]:进出 hypervisor 的额外时间;处理函数的执行时间;TLB/cache miss 造成的时间损失;启用了 EPT(扩展页表)/NPT(嵌套页表)的情况下会造成额外的地址翻译和内存访问开销;启动了 VT-d 或者 IOMMU 的情况下也会造成额外的地址翻译和内存访问开销。这些时间开销使得虚拟机执行同样数目的指令所需的时间与物理机存在差异,这样就产生了基于指令执行差异的计时检测方法。其中,使用最多的指令是 CPUID 和 RDMSR EFER,前者是 Intel VT-x 支持的指令,后者受 AMD SVM 支持,这两者都涉及到硬件虚拟化使能位的读取。通过读取虚拟化使能位,用户如果发现本身关闭了的虚拟化功能被开启了就可以判断存在 HVMR,因此 HVMR 为了实现隐藏会拦截这些指令,通过相应的判断和处理后再将修改后的结果返回给用户,这样便增加了指令执行的时间。以 RDMSR EFER 为例,在 HVMR 截获的情况下其执行时间从 90 个时钟周期增加到 2000 多个[9]。通过多次重复执行该指令,计算总的执行时间差异即可检测出 HVMR 的存在。这种检测方式的可靠性依赖于计时器的可靠性,本地计时器包括[10]、CPU 的时间戳计时器(TSC)、本地时钟、真实计时器、外部计时器。

通过 RDTSC 指令读取 TSC 的值的方法很容易被 HVMR 欺骗。为了更好地支持虚拟化,消减虚拟机与物理机的差异,Intel 和 AMD 硬件虚拟化技术都支持对 TSC 等内部时钟源的拦截,并且支持对 TSC 的修改以使虚拟机读取到的时间值接近真实值。HVMR 可利用这一功能将估计到的时间偏移值从 TSC 上减去然后返回给用户,使其时间区别难以区分出来。

不过这种欺骗方式由于对不同指令的 TSC 的偏移值难以准确估计,其实际效果并不明显,因此 Rutkowska 等人[9]提出使用指令追踪来估计指令运行时间偏移量的方法,该方法获得的 TSC 偏移值较为准确,可以更好地欺骗检测程序。同样的方法也适用于对其他本地计时器的欺骗。此外,基于 TSC 的时间检测方法还有一个问题,就是现代 CPU 的节能管理功能和速度跟踪机制会引起 TSC 值不可信,因此通过这些计时器的检测方法并不适用。考虑到本地时钟和计时器易受到 HVMR 的欺骗,研究人员提出了基于加密 NTP 时钟的检测方法[15],通过加密的网络时钟源检测特定指令的执行时间,根据其与正常系统下的执行时间差异来判断。不过 NTP 时钟的分辨率较低(10 ms 级),这一误差时间内可能执行了上千万条指令,因此要求检测程序运行的时间较长,复杂性也较高,但不失为一种好办法。

针对基于 NTP 等外部时钟的检测方法,BluePill 采用 Blue Chicken 策略[9]。该策略在追踪到同一指令(尤其是 RDMSR EFER 这类指令)重复执行多次时就判定用户在检测 HVMR,从而卸载 hypervisor 进入休眠模式,并且通过钩子技术将代码与内核程序挂钩,在适当的时机再加载运行。不过由于卸载之后的 BluePill 特权级降低容易被检测到,这种反检测方法的实用性有待商榷。

2.2　基于内存资源视图差异的检测方法

硬件虚拟化技术的引进,可能导致内存资源视图出现差异,如 TLB(translation look-aside buffers,旁路转换缓冲)、GDT[16](global descriptor table,全局描述符表)、IDT[17](interrupt descriptor table,中断描述符表)、LDT[18](local descriptor table,本地描述符表)等。其中,对于 TLB 改变的研究最为热门。TLB 用于缓存最近访问的内存,前文提到,HVMR 为了实现隐藏必须截获 CPUID/RDMSR EFER 指令,而这样会引起 TLB 部分或全部的刷新,如果预先填满 TLB,当一条指令要访问 TLB 中本来存在的内存条目时,由于 TLB 中的条目被刷新,就会多一个检索内存的过程,访问内存的时间就会增加,这就是 Ferrie[19]提出的基于特定页表项(PTE)的访问时间变化的检测方法 。不过,因为对 TLB 的访问是随机的,所以很难预先填满所有 TLB 条目,Adams[20]基于 Ferrie 的方法提出的 TLB 检测方法只需要预先填充部分 TLB 条目对应的内存页,通过读取执行 CPUID/RDMSR EFER 指令前后内存页的变化来判断是否存在 BluePill。如果 TLB 条目对应的内存页是原来的页面则证明 TLB 未改变,否则就存在 BluePill。这种方法同时也避免了对时间的测量,因为内部时间不可靠,极有可能遭到 BluePill 的窜改,因此相对于测量访问 TLB 缓存页面的时间的方法更有优势。

但是基于页面内容的 TLB 检测方法对于使用 AMD-V 虚拟化技术的 rootkit 的检测未必适用,这是由于 AMD-V 支持标记 TLB[21],就是对每一个 TLB 条目通过 ASID(地址空间标志符)项来标记它属于主机还是客户,这样就可以避免 hypervisor 刷新客户的 TLB,从而避免被检测。但是实际实现比较困难,也不可能完全不改变 TLB。

不过,rootkit 的设计者可以利用影子页表和嵌套页表来应对这种检测[9],通过影子页表发现用户大量更新 PTE,就可以判断检测状态,从而使用 Blue Chicken 策略实现隐藏。另外,可以通过查找内存中诸如 VMCB. EXITINFO 等信息,看是否属于内核模块来检测 BluePill 的存在,但也很容易被欺骗。这是因为要实现彻底的内存隐藏,BluePill 使用私有页表和 CR3 寄存器[9],将自身的内存资源与客户机的内存资源隔离开来以避免被检测。不过对于这种隔离,检测者可以用嵌套页表来检测 BluePill 的私有页表,但目前并不成熟。

2.3　基于 CPU 异常和错误的检测方法

基于 GP 异常的检测原理是利用 VMsave 指令间接读取 EFER 标志位的值[7]。EFER 只能通过 RDMSR 和 WRMSR 来访问,而这两个指令都会被 hypervisor 截获。检测者为了获取 EFER. SVME 位的值而不被 BluePill 截获,可以使用 VMsave 指令来间接探测。VMsave 指令的功能是保存处理器状态到 RAX 寄存器中物理地址指定的 VMCB。如果 RAX 寄存器中的物理地址是非法地址,该指令在 SVM 未开启时会产生操作码异常(#UD),如果 SVM 开启就会产生通用保护异常(#GP),基于#GP异常的检测原理如下所示:

当 RAX 包含非法物理地址时执行 VMsave 指令:

如果 EFER. SVME = 0,系统产生#UD 异常;

如果 EFER. SVME = 1,系统产生#GP 异常。

因此,VMsave 指令必须能读取 EFER. SVME 的值以判断产生哪种异常,从而可以用来检测 HVBR。但是 BluePill 可以通过设置 VMCB 来截获#GP 异常,截获异常后检查客户 RIP 是否指向 VMsave 指令,如果是而且客户的 EFER. SVME 本来应该为 0,则插入一个#UD 异常欺骗用户。因此为了保证检测的有效性,必须阻止 BluePill 检测客户 RIP 指向的 VMsave 指令。这可以通过 TLB 隐藏检测代码来实现,具体来说就是在执行 VMsave 指令前,检测器先改变检测代码的 PTE,使其指向一个

包含#GP 异常生成代码的假页面,因为检测代码缓存在 ITLB 中,BluePill 找不到,但是 BluePill 可以检测 RIP 的 PTE 地址是否是造的,但是即便发现了伪造,也无法知道产生#GP 异常的指令是什么;如果擅自插入#UD 异常,反而会被发现,此时 BluePill 唯有卸载自己才能避免被检测。

另一种检测方法基于特定 CPU 模型的错误行为(勘误表),如果 HVM hypervisor 不能正确模拟这种错误,检测器就可能检测出来。一个例子是 AMD Erratum 140[22],该错误使得通过 MSR 0x10 读取时间戳计数器 TSC 时,VMCB. TSC_offset 偏移值不会添加到 TSC 上面。但是这种基于 CPU 错误的检测方法依赖于具体的处理器型号,并不通用。

2.4 基于指令计数的检测方法

Barbosa[7]提出的基于指令计数的检测方法本质上也是计时检测方法的一种,不过该方法不依赖于任何时钟,而是通过计数的方法间接地实现指令执行时间的比较,因此不易受到 hypervisor 的欺骗,是目前检测 HVMR 的方法中最简单也最有效的方法。该方法依赖于双核或多核的处理器,其中一个用于执行陷入指令,一个用于简单的循环计数,两者通过 SMP 的共享内存保持同步。通过比较相同的计数次数下陷入指令执行的次数差异来判断有没有 HVMR 的存在,存在 hypervisor 的情况下由于陷入指令的处理消耗了多余的时间,因此指令执行次数会大大减少,差异很明显,如图 4 所示。

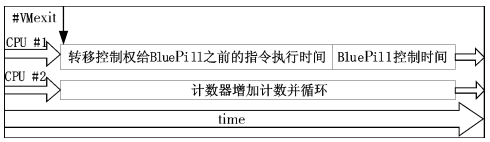

图 4 基于指令计数的检测原理

为了避免检测,BluePill 必须在一开始就阻止计数线程,但是执行#VMexit 一个指令(以停止 BluePill)就要花费 1 000 个时钟周期,发送一个处理器间中断 IPI(以停止 counter)会耗费更多时间,这些时间内检测指令可能已经执行很多次了。通过对比已经可以发现指令执行快慢的明显区别了,因此阻止或者修改计数线程已经太迟了[4]。对于使用外部中断来阻止计数代码执行的方法,检测程序可以通过劫持中断来应对,但是并没必要,因为只需多执行几次检测代码,有一次异常计数就可以判定存在 BluePill 了。

针对 Barbosa 提出的基于陷入指令(如 CPUID)计数的检测方法,Li 等人[23]提出了一种反检测机制,该机制基于计数指令具有的一些特征,通过发送处理器间中断(IPI)识别计数指令,然后修改计数器来欺骗检测程序。文献[23]声称其对于使用 Intel VT-x 技术的 HVMR 反检测有一定作用,使得 HVM 对于虚拟机几乎透明。但是通过分析和验证发现,该方法只适用于对抗静态检测,也就是检测结果在最后显示的情况。对于实时显示指令执行次数的检测,前文已经说明,修改计数线程根本来不及。而且实际使用中由于系统经常有合法的计数指令在执行,不可能对所有的计数指令都进行修改,这样反而会暴露 HVMR 的存在,因此该反检测方法并不实用。

2.5 其他检测方法

a)基于分支预测的检测[7]。分支指令是高度可预测的[24],通过追踪分支指令的执行历史可以预测其将来的行为,分支预测单元使用缓存存储最近的分支指令执行历史。使用静态或动态预测机的规则,可以填充缓存中的历史记录并记录执行代码的时间,检测器执行陷入指令,陷入后的 hypervisor 代码会影响历史记录表,通过时间或者性能计数检测历史页表是否发生了变化从而发现 HVMR。但使用这种检测必须知道分支预测单元 BPU 的实现方式,这个信息很难找到。使用 Milenkovic 等人[25]开发的 BPU 可以实现,但很困难。

b)利用最后一条跳转 LBR 记录的检测[19]。因为 SVM 支持 LBR 虚拟化而失效;利用 L1 和 L2 数据/指令缓存的检测[26],hypervisor 可以通过禁用该缓存使其失效。

2.6 HVMR 存在性检测方法小结

表 3 总结了以上 HVMR 检测方法的优点和缺点,从中可以看出,在实现复杂度上基于指令执行时间差异的方法最为简单,但同时也最容易被欺骗,可靠性最低。基于内存资源视图差异与基于 CPU 异常和错误的检测方法有效性相对较高,但是依赖于具体的硬件虚拟化类型和 CPU 型号,通用性不高。相比之下,基于指令计数的检测方法,既继承了基于指令执行时间检测方法实现简单的优点,又不依赖于具体的时钟,而是通过充分发挥多核 CPU 并行处理的优点,实现比较计数,从而发现指令执行时间的差异,是当前最适用的 HVMR 存在性检测方法。虽然其只适用于两个以上核心的 CPU,但多核 CPU 架构已然成为当前计算机的主流配置,因此并不影响其通用性。

表 3 HVMR 存在性检测方法比较

检测方法	优点	缺点
基于指令执行时间差异的检测方法	可用的时钟源较多,虚拟化引入的时间开销体现在很多方面,易于实现	时钟源很容易被 HVMR 截获和修改,准确性不可靠
基于内存资源视图差异的检测方法	可以实现不依赖于时钟源,并可结合嵌套页表技术实现对深度隐藏的 BluePill 的检测	对于使用 AMD-V 虚拟化技术的 rootkit 不适用,实现较为复杂
基于 CPU 异常和错误的检测方法	检测基于 CPU 固有的异常和错误,可绕过 hypervisor 的拦截读取 EFER,很难被反检测	检测方法依赖于具体的处理器型号,并不通用
基于指令计数的检测方法	不依赖于任何时钟,通过计数的方法来间接地比较指令的执行时间,不易受到 hypervisor 的欺骗	只适用于双核或者多核 CPU 架构的检测

3 HVMR 恶意性检测与预防

上述 HVMR 存在性检测方法都有一个前提,即不存在嵌套的虚拟化环境(VME)或 hypervisor,也就是说系统没有使用虚拟化的条件下若检测出 VME,就可以判断该 VME 是非法的 HVMR,否则无法判断检测出来的 VME 是否合法。尤其是在未来虚拟化技术普遍应用的情况下,光是检测出 VME 没有多大现实意义,因此如何识别检测到的 VME 是否恶意是未来的重点研究方向。最终的检测方法还是要落到 HVMR 在内存中的代码的特征分析上来,由于 HVMR 采用了私有页表、指令拦截等隐藏措施,使得在虚拟机层次的检测显得困难重重,这本质上是因为合法的系统被虚拟化之后特权级比 HVMR 低所导致的,因此要想有效地检测和预防 HVMR,首先需取得比 HVMR 高或者至少同等的特权,以获取对于内存的完全访问,从而分析内存中是否有 HVMR 的特征。为此,本文在以上研究的基础上提出了基于 hypervisor 的 HVMR 恶意性检测方法和基于硬件的 HVMR 恶意性检测方法。

3.1 基于 hypervisor 的检测方法

基于 hypervisor 的检测方法,通过提前加载带检测模块的 hypervisor 或者修改现有的合法 hypervisor 加入检测代码,检测

任何企图访问硬件虚拟化相关资源的代码,并对相应资源实行写保护,可以防止 HVMR 的加载,起到预防作用,也可以用来对付 BluePill 的 Chicken 策略,防止休眠的 BluePill 复活。对于已经存在 HVMR 的情况,再次加入检测 hypervisor 需要硬件支持嵌套虚拟化,即检测 hypervisor 和加载 rootkit 的恶意 hypervisor 并存,这种情况下由于两者的特权级相同,其检测与反检测之间的博弈和对抗类似于内核 rootkit 的检测与反检测技术,没有绝对可靠的检测方法。

为了读取 BluePill 的私有内存并分析特征,必须获取对私有内存的完全访问。最简单的办法是通过修改特定的虚拟地址的页表项(PTE)以指向想要访问的物理地址,但 PTE 修改法容易遭到恶意 hypervisor 的识别并使用影子页表触发页面错误#PF,或者使用嵌套页表阻止客户修改 hypervisor 代码页,而且得到的内存空间是分散的。因此,与其检测 BluePill 代码特征,不如扫描 VMCB/VMCS 的特征,其中包含 BluePill 的信息,如 CR3 寄存器值等,通过 CR3 寄存器可以访问页表,从而看到连续的虚拟内存空间。一般情况下,合法的 hypervisor 只要检测到多个 VMCB/VMCS 结构就可以判定多余的 VMCB/VMCS 结构是因 HVMR 而增加的,从而检测出系统受到感染。

3.2　基于硬件的检测方法

基于硬件的内存获取方法主要是利用 DMA(direct memory access,直接内存访问)技术让设备直接访问内存空间,从而获取 hypervisor 的代码并分析,这类硬件包括 FireWire 总线、部分 PCI 设备(如 Tribble、CoPilot)、GART(图形地址重映射表)等。但是由于 IOMMU 和 VT-d 等技术的出现,新的 CPU 支持对 DMA 的内存映射,而不是直接访问。另外,通过修改北桥内存映射的 I/O 地址分配表可以禁用除 CPU 之外的其他设备对内存的直接访问,试图访问原始内存甚至会导致死机,因此基于 DMA 的方法不再适用[10]。

不过通过精心设计内存控制器,可以允许特定的外部访问[27]。而且没必要专门增加这类硬件,通过芯片组中现有的可编程植入式微控制器(embedded μcontroller)就能实现利用 DMA 访问内存并扫描 rootkit,如图 5 所示。可编程植入式微控制器是 Intel 芯片用来执行固件的数字签名的,其结果存储在不可变的串行闪存中。通过对其编程,可以用来访问 DRAM 内存,然后搜索内存中与 Flash 中存储的 rootkit 特征匹配的代码从而检测到 HVMR。而且相比利用 PCI 硬件的方法,芯片组固件可以通过 DMA CTRL 访问并关闭 DMA 重映射单元以方便检测程序运行,这样就能比 hypervisor 更加接近硬件。但是检测程序与具体的芯片组类型相关。另外一种类似的方法是通过系统管理模式(SMM,可以直接访问内存,控制底层硬件)运行检测程序以扫描内存,但 SMM 也可能被 rootkit 开发者利用。限于篇幅,本文在此不作具体讨论。

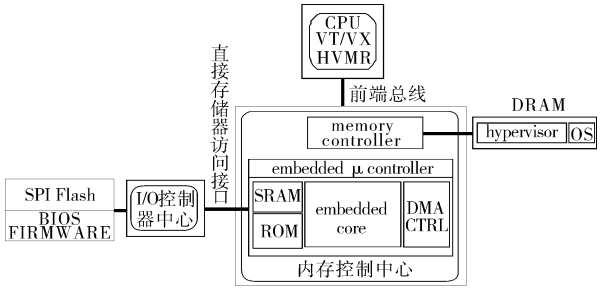

图 5　基于硬件的 rootkit 检测

4　结束语

本文在分析硬件虚拟化技术和硬件虚拟化 rootkit(HVMR)的基础上,总结了普通 rootkit 检测方法在检测 HVMR 时的局限性,分析了针对 HVMR 的检测技术,包括基于指令执行时间差异的检测、基于内存资源试图差异的检测、基于 CPU 异常和错误的检测、基于指令计数的检测等,并总结了各种方法的优缺点。通过分析其有效性和对应的反检测方法,表明这些检测方法虽然可以检测出硬件虚拟化 hypervisor 的存在,但并不能证明这些 hypervisor 的恶意性。为此提出了基于 hypervisor 的恶意性检测方法和基于硬件的恶意性检测方法,前者通过对涉及硬件虚拟化的关键资源进行写保护来阻止 HVMR 的加载和运行,而后者通过对内存代码进行扫描发现恶意 HVMR 的特征。

随着虚拟化技术的普及,仅仅是检测出存在硬件虚拟化环境并不能判断恶意 rootkit 的存在,因为合法的系统也将大量使用硬件虚拟化技术。检测和反检测技术的博弈将在新的特权级层(hypervisor 层)继续进行,虚拟化安全将继续得到巩固。未来针对 HVMR 的对策将主要是基于合法 hypervisor 的预防为主、基于硬件的检测为辅的预防和治理体系。另外,将可信计算与虚拟化结合的技术将为防治 HVMR 提供新的思路[28]。

参考文献:

[1] KING S T, CHEN P M, WANG Yi-min, *et al*. SubVirt: implementing malware with virtual machines[C]//Proc of IEEE Symposium on Security and Privacy. Washington DC: IEEE Computer Society, 2006: 314-327.

[2] RUTKOWSKA J. Subverting Vista™ kernel for fun and profit[C]//Proc of Symposium on Security for Asia Network . New York: Matasano Security, 2006.

[3] ZOVI D Z. Hardware virtualization rootkits[C]//Proc of Symposium on Security for Asia Network. New York: Matasano Security, 2006.

[4] ABRAMSON D, JACKSON J, MUTHRASANALLUR S, *et al*. Intel® virtualization technology for directed I/O(VT-d)[J]. Intel Technology Journal, 2006, 10(3): 179-192.

[5] 凌冲,孙乐昌,刘京菊. 一种硬件虚拟化技术的 rootkit 及其检测[J]. 西安科技大学学报,2010,30(1):86-91.

[6] 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.

[7] BARBOSA E. Detecting BluePill[C]//Proc of Symposium on Security for Asia Network, 2007.

[8] PTACEK T, LAWSON N, FERRIE P. Don't tell Joanna' the virtualized rootkit is dead[R]. New York: Matasano Security, 2007.

[9] RUTKOWSKA J, TERESHKIN A. IsGameOver(), anyone? [C]//Proc of Black Hat Conference. 2007.

[10] FRITSCH H. Analysis and detection of virtualization-based rootkits [D]. [S.l.]: Technische Universitat Munchen, 2008.

[11] NARAINE R. Rutkowska faces '100% undetectable malware' challenge[EB/OL]. (2007-06-27). http://www.zdnet.com/blog/security/rutkowska-face-100-undetectable-malware-challenge/334.

[12] BULYGIN Y. CPU side-channels vs. virtualization rootkits: the good, the bad, or the ugly[C]//Proc of TOORCON. 2008.

[13] NING Jian, WANG Huai-tmin, GUO Shi-ze, *et al*. CBD: a counter-based detection method for VMM in hardware virtualization technology [C]//Proc of the 1st International Conference on Pervasive Computing Signal Processing and Applications. Washington DC: IEEE Computer Society, 2010: 352-356.

(下转第 12 页)

object localization with superpixel neighborhoods[C]//Proc of the 12th International Conference on Computer Vision. Washington DC: IEEE Computer Society, 2009: 670-677.

[21] WANG Shu, LU Hu-chuan, YANG Fan, *et al*. Superpixel tracking [C]//Proc of IEEE International Conference on Computer Vision. 2011:1323-1330.

[22] ZHOU Xue, LI Xi, CHIN T J, *et al*. Superpixel-driven level set tracking[C]//Proc of the 19th IEEE International Conference on Image Processing. 2012:409-412.

[23] LIU Li-wei, XING Jun-liang, AI Hai-zhou, *et al*. Semantic superpixel based vehicle tracking[C]//Proc of the 21st International Confe-rence on Pattern Recognition. Berlin:Springer-Verlag, 2012:2222-2225.

[24] WANG Wei-jun, NEVATIA R. Robust object tracking using constellation model with superpixel[C]//Proc of the 11th Asian Conference on Computer Vision. Berlin:Springer-Verlag,2012:191-204.

[25] HOIEM D, EFROS A A, HEBERT M. Automatic photo pop-up[J]. ACM Trans on Graphics, 2005, 24(3):577-584.

[26] MAJI S, VISHNOI N K, MALIK J. Biased normalized cuts[C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. Washington DC:IEEE Computer Society, 2011:2057-2064.

[27] MOORE A, PRINCE S. Scene shape priors for superpixel segmentation[C]//Proc of the 12th IEEE International Conference on Computer Vision. Washington DC:IEEE Computer Society,2009:771-778.

[28] MOORE A, PRINCE S J D, WARRELL J. Lattice cut -constructing superpixels using layer constraints[C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2010:2117-2124.

[29] DIGABEL H, LANTUEJOUL C. Iterative algorithms[C]//Proc of the 2nd European Symposium on Quantitative Analysis of Microstructures in Material Science, Biology and Medicine. Sturrgart: Riederer-Verlag, 1978.

[30] FUKUNAGA K, HOSTETLER L. The estimation of the gradient of a density function, with applications in pattern recognition[J]. IEEE Trans on Information Theory, 1975, 21(1): 32-40.

[31] CHENG Yi-zong. Mean shift, mode seeking, and clustering[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1995, 17(8): 790-799.

[32] WANG Jue, THIESSON B, XU Ying-qing, *et al*. Image and video segmentation by anisotropic kernel mean shift[C]//Proc of European Conference on Computer Vision Berlin: Springer-Verlag,2004:238-249.

[33] TAO Wen-bing, JIN Hai, ZHANG Yi-ming. Color image segmentation based on mean shift and normalized cuts[J]. IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics, 2007, 37(5):1382-1389.

[34] XIANG Shi-ming, PAN Chun-hong, NIE Fei-ping, *et al*. Turbopixel segmentation using Eigen-images[J]. IEEE Trans on Image Processing, 2010, 19(11): 3024-3034.

[35] CIGLA C, ALATAN A A. Efficient graph-based image segmentation via speeded-up turbo pixels[C]//Proc of the 17th IEEE International Conference on Image Processing. 2010:3013-3016.

[36] REN C Y, REID I. gSLIC: a real-time implementation of SLIC superpixel segmentation[R]. Qxford: Department of Engineering, University of Oxford,2011.

[37] LUCCHI A, LI Yun-peng, SMITH K, *et al*. Structured image segmentation using kernelized features[C]//Proc of the 12th European International Conference on Computer Vision. Berlin:Springer-Verlag, 2012: 400-413.

[38] BOYKOV Y Y, JOLLY M P. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images[C]// Proc of the 8th IEEE International Conference on Computer Vision. 2001: 105-112.

[39] GU Guang-hua, LI Feng-cai, ZHAO Y, *et al*. Scene classification based on spatial pyramid representation by superpixel lattices and contextual visual features[J]. Optical Engineering, 2012, 51(1): 017201-1-017201-8.

[40] TIGHE J, LAZEBNIK S. Superparsing: scalable nonparametric image parsing with superpixels[C]//Proc of the 11th European Computer Vision. Berlin: Springer-Verlag, 2010: 352-365.

(上接第5页)

[14] DESNOS A, FILIOL E, LEFOU I. Detection of an HVM rootkit (aka BluePilllike)[J]. Computer Virology, 2011, 7(1):23-49.

[15] KYTE I, ZAVARSKY P, LINDSKOG D. Enhanced side-channel analysis method to detect hardware virtualization based rootkits[C]// Proc of World Congress on Internet Security. 2012:192-201.

[16] RUTKOWSKA J. Red Pill or how to detect VMM using (almost) one CPU instruction [EB/OL]. (2004-11-17). http://www.securiteam.com/securityreviews/6ZooH20BQS.html.

[17] RUTKOWSKA J. Blue Pill detection [EB/OL]. (2006-03-12). http://theinvisiblethings.blogspot.com/2006/08/blue-pill-detection.html.

[18] QUIST D, SMITH V. Detecting the presence of virtual machines using the local data table [EB/OL]. http://index-of.es/Misc/vm.pdf.

[19] FERRIE P. Attacks on virtual machine emulators [EB/OL]. http://www.symantec.com/avcenter/reference/Virtual_Machine_Threats.pdf.

[20] ADAMS K. BluePill detection in two easy steps [EB/OL]. (2007-07-02). http://x86vmm.blogspot.com/2007/07/bluepill-detection-in-two-easy-steps%.html.

[21] Virtualization technology for AMD architecture [EB/OL]. http://download.microsoft.com/download/9/8/f/98f3fe47-dfc3-4e74-92a3-088782200fe7/twar05014_winhec05.pdf.

[22] MYERS M, YOUNDT S. An introduction to hardware-assisted virtual machine (HVM) rootkits[R]. [S.l.]:WhitePapers DB. 2007.

[23] LI He-shuai, ZHU Jun-hu, ZHOU Tian-yang, *et al*. A new mechanism for preventing HVM-aware malware [C]//Proc of the 3rd International Conference on Communicution Software and Networks. 2011: 163-167.

[24] SHEN J, LIPASTI M. Modern processor design: fundamentals of superscalar processors [M]. [S.l.]: McGraw-Hill, 2005.

[25] MILENKOVIC M, MILENKOVIC A, KULICK J. Demystifying Intel branch predictors [EB/OL]. http://www.ece.wisc.edu/~wddd/2002/final/milenkovic.pdf.

[26] FERRIE P. Attacks on more virtual machine emulators [EB/OL]. (2007-08-31). http://pferrie.tripod.com/papers/attacks2.pdf.

[27] BULYGIN Y, SAMYDE D. Chipset based approach to detect virtualization malware[R]. [S.l.]:Intel Corporation, 2008.

[28] GARFINKEL T, PFAFF B, CHOW J, *et al*. Terra:a virtual machine-based platform for trusted computing [C]//Proc of the 19th ACM Symposium on Operating Systems Principles. New York:ACM Press, 2003:193-206.

[29] 于淼,威正伟. NewBluePill 深入理解硬件虚拟机[M]. 北京:清华大学出版社,2011:55.

[30] WMware. Timekeeping in VMware virtual machines [EB/OL]. http://www.vmware.com/pdf/vmwaretimekeeping.pdf.