一种基于完整性保护的终端计算机安全防护方法

李清宝 张 平 曾光裕
(国家数学工程与先进计算重点实验室 郑州 450000)

摘 要 终端计算机是网络空间活动的基本单元,其安全性直接关系着网络环境和信息系统的安全。提出了一种基于完整性保护的终端计算机安全防护方法,它将完整性度量和实时监控技术相结合,保证终端计算机运行过程的安全可信。建立了以TPM为硬件可信基、虚拟监控器为核心的防护框架,采用完整性度量方法建立从硬件平台到操作系统的基础可信链;在系统运行过程中监控内核代码、数据结构、关键寄存器和系统状态数据等完整性相关对象,发现并阻止恶意篡改行为,以保证系统的完整、安全和可靠。利用Intel VT硬件辅助虚拟化技术,采用半穿透结构设计实现了轻量级虚拟监控器,构建了原型系统。测试表明,该方法能够对终端计算机实施有效的保护,且对其性能的影响较小。

关键词 终端计算机,完整性,虚拟监控器,完整性度量,完整性监控
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.6.035

Integrity Based Security Protection Method for Terminal Computer

LI Qing-bao ZHANG Ping ZENG Guang-yu
(State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450000, China)

Abstract Terminal computer is the basic unit of network activities, which is directly related to the security of network environment and information systems. An integrity based security protection method for terminal computer was proposed, which integrates integrity measurement and real-time monitoring technology to ensure the security and credibility of terminal computer. A protection framework was established, which uses TPM as hardware trusted base and virtual monitor as the core unit. Integrity measurement is used to establish the basic trusted chain from the hardware platform to operating system. And integrity related objects, such as kernel code, data structures, key registers and system status data, are monitored when the system is running to detect and prevent from malicious tampering in order to ensure system integrity, security and reliability. A lightweight virtual machine monitor was designed using Intel VT hardware-assisted virtualization technology and a prototype system was realized. Tests show that the method is effective and has less impact on the performance of terminal computer.

Keywords Terminal computer, Integrity, Virtual machine monitor, Integrity measurement, Integrity monitoring

1 概述

随着互联网技术的发展,网络终端计算机的安全问题日趋严重。作为网络的重要组成部分,终端计算机存储和处理大量的个人和单位用户的重要数据,但是,相比于网络服务器,终端计算机的管理和安全防护并未受到足够的重视,越来越多的终端计算机成为各种网络安全攻击和破坏的目标。所以,终端计算机的安全防护对于保证其本身的信息安全和网络安全都是非常重要的。目前网络终端计算机常用的操作系统(Windows和Linux)变得越来越庞大,各种安全漏洞不可避免[1],所以建立在操作系统之上的各种安全防护技术(如病毒检测、木马查杀、网络防火墙和入侵监测等)不能从根本上解决问题,因此,要从根本上保证终端计算机的安全,防护系统必须与操作系统相隔离,首先保证操作系统本身的安全可信。

各种恶意软件和攻击手段都需要通过非法篡改系统对象来实现破坏行为和隐藏行为轨迹,如修改系统调用、中断调用、修改内核数据结构、系统服务程序等,即都会影响和破坏系统的完整性,从理论上讲可以通过监控和检测系统的完整性来保护终端计算机的安全。只要能及时发现破坏系统完整性的行为,就可以有效避免恶意软件对系统的攻击和破坏;只要能够保证系统的完整性不被破坏,就可以有效保护系统安全。由于完整性监控不需要像现有检测和防御软件那样依赖于已知的恶意软件特征和行为规则,因此基于完整性保护的安全防护技术不仅能防御已知恶意软件,对未知攻击同样有效。

完整性保护主要有检测和监控两种技术。完整性检测又称完整性度量,即通过提取系统完整性相关对象的度量特征值来验证其完整性是否遭到破坏;完整性监控则是监控系统软件的操作行为,发现是否存在影响系统完整性的行为,并采

到稿日期:2014-07-15 返修日期:2014-10-27 本文受核高基专项资助。
李清宝(1967—),男,博士,教授,CCF高级会员,主要研究方向为信息安全和系统结构,E-mail:psx111@tom.com;张 平(1969—),女,副教授,主要研究方向为信息系统安全;曾光裕(1964—),女,副教授,主要研究方向为计算机应用。

取相应策略保护系统的完整性。由于完整性检测过程复杂，实施困难，对网络终端的性能影响也较大，因此完整性检测较多采用了静态或周期性检测的方法实现，存在 TOC-TOU (Time-of-Check to Time-of-Use)问题[2]。完整性监控可以分为应用层监控、系统层监控、虚拟层监控和硬件监控。应用层监控和系统层监控实现相对简单，从而较多地被采用，但缺点是容易被获得高级系统权限的恶意代码所旁路、禁用或篡改，失去监控功能；硬件层监控因开发难度高和成本开销大而较少被使用。虚拟机技术较好地解决了开发者直接面对硬件电路开发的技术难题。虚拟机位于操作系统层和硬件层之间，具有很好的隔离性和可干预性，不足之处是缺乏上层系统的语义信息，需要针对不同的监控目标构建目标语义。

基于上述分析，本文提出一种基于操作系统完整性保护的思想来解决网络终端安全的防护方法，主要工作包括：

(1)基于完整性保护思想，建立一个从可信硬件→虚拟监控器→操作系统→应用软件的完整的安全防护体系。利用 TPM 和完整性度量与验证机制，保证系统初始态的安全可信。

(2)基于硬件虚拟化技术实现一个轻量级的虚拟监控器 LHvisor，它和操作系统紧耦合，支持对影响系统完整性的操作和行为进行实时监控，保证系统运行过程的安全可信。

(3)基于虚拟机自省技术，在虚拟监控器 LHvisor 中建立系统完整性相关对象语义信息，实现对恶意行为主体的追踪和隐藏对象的发现。

(4)设计实现了原型系统 IBTPS(Integrity Based Terminal Protection System)，验证了方法的有效性，并对系统性能的影响进行了测试分析。

2 操作系统完整性分析

操作系统作为终端计算机的核心系统软件，负责管理计算机的各种软硬件资源和向用户提供各种服务。操作系统的完整性体现在能够按照设计的方式对各种事件和请求做出正确、快速的响应和处理，即操作系统执行预期的行为并给出符合预期的结果。操作系统完整性被破坏表现为系统运行行为和功能发生改变。为实现对操作系统完整性的有效保护，首先要确定系统中与完整性相关的对象，即内核中哪些对象是恶意攻击破坏的目标，哪些行为会破坏内核完整性。

基于对各种队操作系统的攻击行为的分析，可以将系统完整性相关对象分为两大类：控制类和数据类。

(1)控制类

操作系统执行的控制流决定系统的行为和实现的功能。各种攻击为实现其恶意功能通常会在内核中植入恶意代码或篡改系统控制相关结构，改变系统正常的执行路径，执行攻击者设定的功能或行为，即破坏操作系统的控制完整性。完整性相关的控制类对象主要包括：

• 内核代码：系统功能由代码的执行来实现，因此篡改内核代码是改变系统执行路径、破坏操作系统完整性最直接的方法，如典型的 rootkit(enyelkm、wnps 和 kbeast 等)就采用这种方式。篡改代码常以调用函数劫持的方式实现，如篡改中断服务程序代码或系统调用函数的开始几条指令，跳转去执行其恶意代码[3]。

• 内核控制相关数据结构包括中断向量表、系统调用表、全局描述符表等，这些数据结构的内容控制系统执行的路径，对这些数据结构的篡改是内核攻击最常采用的方法[4]。如 all-root、ipsecs-kbeast-v1 、override、Rkit 、Synapsys 及 adore 等 rootkit 都是通过修改相关数据结构来实现其恶意功能的。

• 控制相关寄存器包括中断描述符表寄存器(Interrupt Descriptor Table Register，IDTR)和全局描述符表寄存器(Global Descriptor Table Register，GDTR)。IDTR 和 GDTR 分别存放中断描述符表(Interrupt Descriptor Table，IDT)和全局描述符表(Global Descriptor Table，GDT)的首地址。在系统运行过程中，恶意攻击可通过修改 IDTR 或 GDTR 的值使其指向攻击者预设的新描述符表，从而实现对系统控制流的篡改，破坏系统完整性。

(2)数据类

数据类完整性相关对象指操作系统运行时所建立和维护的系统数据结构和状态数据，将其完整性称为操作系统的数据完整性[5]。数据完整性对操作系统来说同样重要，操作系统的服务行为要依赖于相关的数据对象，服务功能效果也反映在这些相关的数据对象上，例如进程控制块、页表、文件索引节点等。这些数据对象可以被系统服务程序用来向管理员和用户提供系统运行的各种状态信息，如进程状态、文件系统信息、网络连接状态等。这些数据对象同样可以被恶意代码(如 rootkit)所利用，攻击者通过篡改、过滤这些系统数据，伪造虚假的底层系统信息给应用层从而达到隐蔽感染痕迹、避免被查杀、维持持久后门等目的，如进程隐藏、网络连接隐藏、模块隐藏等，为进一步实现攻击建立基础。

3 完整性保护的实现原理

对操作系统完整性的保护通过保护完整性相关对象来实现。根据不同对象的不同特性采用相应的保护措施。

3.1 内核代码保护

操作系统功能由内核代码的执行来实现。内核代码的存在方式主要有两种状态：存储状态和运行状态。操作系统内核代码一般以文件形式存储于存储介质中，系统启动后被 loader 程序加载到内存空间，其完整性需要在加载前、加载中和加载后的整个运行过程中被保证。首先，要保证操作系统加载前是完整可信的；其次要保证系统引导过程的可信；即加载到内核空间的代码是完整的；再次要保证操作系统代码在运行过程中是完整可信的。所以对内核代码的保护采用完整性度量和实时监控相结合的方式来实现，具体讲就是在系统加载前对存储在介质中的操作系统文件进行完整性度量，保证内核的静态完整性；在加载过程中保证加载到内存中的代码是完整的；在运行过程中对代码段所在内存页面进行访问控制保护，采用行为监控方式阻断对代码的恶意篡改行为，保护内核的动态完整性。

(1)静态完整性保护

静态完整性通过系统可信引导和可加载内核模块 LKM 的加载时度量来保证。根据 TCG 的可信计算规范，可信引导采用链式度量方式[6]，建立硬件→BIOS→基本内核的基础信任链，保证基本内核代码的静态完整性。LKM 的加载会造成内核的演变，所以在 LKM 加载时进行完整性度量，以保证加载的 LKM 代码的静态完整性。

(2)动态完整性保护

在系统运行过程中同样要防止对内核代码的恶意篡改。操作系统通常也提供了访问控制保护：通过将内核代码所在

页面设置为不可写，防止对内核代码的修改。但是恶意攻击可以在获得系统访问的超级权限后，关闭写保护机制，修改内存页的访问权限，实现对代码的恶意篡改。如 x86 架构中，页保护机制的开启和关闭是由控制寄存器 CR0 的 WP 位控制，当 CR0. WP=0 时，页保护机制关闭；CR0. WP=1 时，页保护机制开启。因此，如果攻击者获得权限并关闭页保护机制，即将 CR0. WP 置零，则可以实现对内核代码页的篡改。

为了避免这种攻击，在虚拟机层实现对内核代码所在内存页的访问控制保护，由基于 Intel VT 提供的 EPT 内存虚拟化技术实现[7]。通过将内核代码所在内存页对应的 EPT 页表项中的属性设置为只读，客户机中超级用户对内存页的访问权限除了受到 CR0. WP 和客户机中页表属性的限制外，还要受到 EPT 页表项中的属性约束。因此，即使攻击者在获取超级用户权限并将 CR0. WP 置零或者修改客户机页表属性后，超级用户也无权对 EPT 页表项中被设为只读的内存页进行写操作。另外，由于只有虚拟监控器能够对 EPT 页表项进行操作，攻击者要实现对 EPT 页表项中的属性修改需要获取更高的权限，实现难度更大，因此该方法能实现对内核代码页的有力保护。

3.2 控制结构和控制寄存器保护

系统调用表、中断描述符表和全局描述符表等数据结构中包含了同内核功能实现紧密相关的控制信息，如系统调用函数入口地址、中断服务程序入口地址等，这些内容的改变将使系统的执行路径发生改变，从而改变系统的预期行为，破坏系统完整性。而这些数据结构的特征在操作系统编译和加载后及系统运行过程中，其物理地址和内容不再发生改变，且所占空间较大，所以可以通过对其所在内存页的读写访问控制机制，实现对其完整性的监控和保护。在 Linux 环境下其地址范围可以通过分析系统符号表文件 System. map 获得。

对控制寄存器最有效的保护是监控对其访问行为，截断非法修改。在 x86 架构中，实现对 GDTR 和 IDTR 修改的指令分别为 LGDT 和 LIDT，但正常情况下，IDTR 和 GDTR 的值在客户操作系统引导完成之后就不再发生变化，故此后对它们的所有修改都可视为非法操作。

为保护 IDTR 和 GDTR，使虚拟监控器从系统启动完成之时开始对 LIDT 和 LGDT 指令进行监控，当捕获到它们执行时，产生 VM exit，陷入到虚拟监控器中进行处理，根据退出原因编号和退出信息域中的信息判断出陷入原因为执行 LIDT 或 LGDT 指令时，对客户机中的 IP 执行加 1 操作，然后退出虚拟监控器，使客户机跳过该指令继续执行，从而保证系统运行过程中 GDTR 和 IDTR 的值不被篡改。另外，在实现保护的同时，将所有意图修改 GDTR 和 IDTR 的事件信息记入监控日志。

3.3 数据完整性对象保护

恶意攻击往往通过篡改数据完整性对象，向用户提供虚假信息，欺骗检测和防御软件，来实现其隐藏自身行为的目的，但是这些隐藏对象若要在系统运行过程中发挥作用，它们在系统底层和执行相关的信息和数据就不能被修改，这样就造成了隐藏对象的相关内核数据不一致的情况，即破坏了系统逻辑关系完整性，而这种不完整性也为这些隐藏对象的检测、发现和清除提供了方法：分别构造系统内底层和高层视图，通过交叉视图对比，验证数据逻辑关系完整性[8]。

数据完整性保护通过对数据关联对象操作行为的监控建立真实的对象底层状态视图，再通过常用方法获取高层状态视图，然后通过对比底层和高层状态值，检测底层数据和高层数据的逻辑关系完整性是否被破坏，进而发现隐藏对象。通常关注的数据完整性对象主要有：进程、模块、文件和网络连接等，本文的讨论主要以进程相关数据完整性的监控和检测为例，其它数据对象的实现原理相同。实现数据完整性保护的关键是如何获得真实的底层信息并将其转换成具有系统语义的高层信息。

4 终端计算机安全防护框架

基于完整性保护的思想，我们采用虚拟机监控技术设计了终端计算机安全防护框架，并基于 Intel VT 实现了原型系统 IBPST。

4.1 框架结构

终端计算机安全防护框架综合考虑各种影响系统完整性的因素，保证终端从启动到运行的全过程安全可信，其框架结构如图 1 所示。

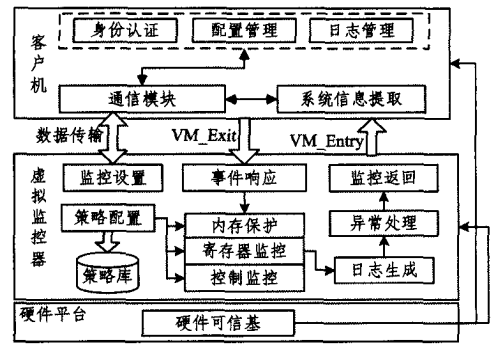

图 1 基于完整性保护的终端安全防护框架

防护框架以可信平台模块 TPM(Trust Platform Module)为硬件可信基，以完整性度量方式保证系统各组件(包括虚拟监控器)的基础可信性；以虚拟监控器为核心，对系统运行时的完整性进行动态保护。虚拟监控器位于操作系统之下，可以在操作系统加载前启动，也可在系统运行过程中动态加载和关闭。虚拟监控器对预先配置对象进行监控，当监控事件发生时客户机陷入到虚拟监控器，对事件进行处理。虚拟监控器中事件响应模块对陷入事件进行分析，根据事件类型调用相应的事件处理模块(包括内存保护模块、寄存器监控模块和事件监控模块)处理，事件处理模块依据策略库中的处理策略对事件进行处理，如发现异常事件，则记录日志，并提交异常处理模块处理，处理结束陷出监控器返回客户机。客户机和虚拟监控器间通信由通信模块完成，主要完成配置信息、策略信息和高层数据视图的传递。

4.2 防护流程

图 2 给出了基于完整性保护的终端计算机安全防护流程。首先建立初始可信环境，保证系统的基础运行环境安全可信。基于 TCG 的可信引导过程，以硬件可信平台模块 TPM 作为基础可信基，采用链式度量方式建立从 TPM 到操作的系统初始可信链，由于引入了虚拟监控器，因此可信链调整为硬件→BIOS→OS loader→虚拟监控器→OS loader→基本内核：由 OS loader 度量虚拟监控器并启动虚拟监控器后，再度量并启动操作系统基本内核；然后在系统运行过程中通过动态监控完整性相关对象，及时发现、处理和报告破坏系统完整性的各种操作和行为，并采用多视图对比的方法验证数

据对象的逻辑完整性，发现和清除系统内隐藏的恶意对象，从而实现终端系统整个生命期的安全可信。

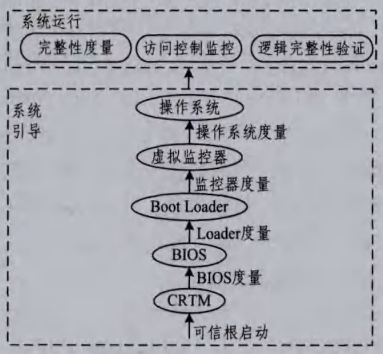

图 2　基于完整性保护的终端计算机安全防护流程

4.3　关键技术实现

4.3.1　Intel VT 虚拟监控器设计

虚拟监控器是终端计算机安全保护的核心，设计和实现虚拟监控器必须考虑的两个关键要素是性能和安全性。目前，虚拟监控器主要基于已有的虚拟机系统（如 Xen 和 KVM 等）来实现[8,9]，存在性能损失大、可信基膨胀及安全隐患等问题[10]。本文从保护终端计算机安全的角度出发，采用 Intel VT 硬件辅助虚拟化技术设计了一个单客户机紧耦合的轻型虚拟监控器。

Intel VT 是 Intel 提出的 x86 平台硬件辅助虚拟化解决方案，为虚拟机监控器提供了全新的特权空间，并通过硬件底层指令集的扩展实现了对虚拟监控器和客户机之间进行切换的支持，提高了虚拟化技术的性能和可靠性。Intel VT 以硬件辅助方式实现了对内存虚拟化、CPU 虚拟化和 IO 虚拟化的支持，利用这些技术可实现对完整性相关对象的有效保护。

LHVisor 借鉴了 BitVisor 半穿透的设计思想，基于 Intel VT 技术实现，直接运行在硬件层上，不需依附于主操作系统和管理域（如 Xen 的 Domain0），仅支持一个客户操作系统，并与之紧密耦合，监控器内仅实现对完整性相关操作和事件的截获与处理，其它资源管理、I/O 等系统控制和处理仍由操作系统完成。LHVisor 的实现原理如图 3 所示。

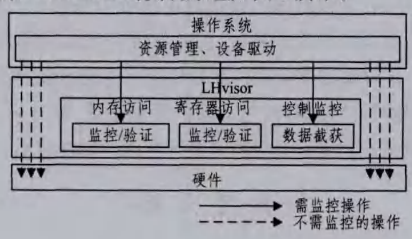

图 3　LHVisor 实现原理图

4.3.2　内核空间访问监控

内核代码的完整性保护采用 Intel VT 的内存虚拟化——扩展页表 EPT 内存保护机制实现。首先通过对 Linux 系统导出的符号表 System. map 的分析，获得内核代码的虚拟地址空间范围，再计算出相应的物理地址空间范围。采用动态构建 EPT 页表的方式对该段内存空间进行保护：当虚拟监控器在进行内存初始化时，每一个内存页第一次被访问时都会发生 EPT Violation 异常，陷入虚拟监控器，虚拟监控器捕获异常并分析异常原因，如果是由于页表项不存在（EPT 页表项的低三位全为零）触发的异常，则获取客户机陷入时的内存地址，并构建该地址所对应的内存页的 EPT 页表项。在给页表项赋值时，首先判断该页是否为内核代码所在内存页：如果是，则设置该页表项的 w 位为零；否则，正常设置。通过上述机制，在 EPT 中将内核代码内存页设置为只读。在系统运行过程中，发生对内核代码内存页的写操作时，就会因为违背内存页的读写属性而发生 EPT Violation 异常，然后在虚拟机监控器中对事件进行相应处理，如跳过指令执行，使对该页的修改操作失败，从而实现了对内核代码完整性的有效保护。上述方法实现原理如图 4 所示。

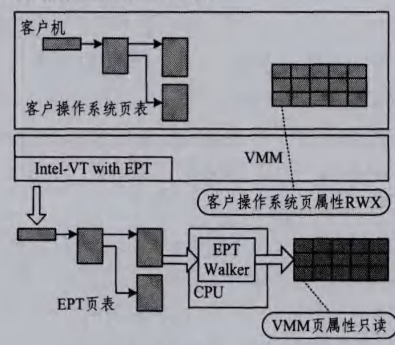

图 4　基于 EPT 的内存页只读保护机制

4.3.3　控制寄存器监控

控制寄存器的监控通过对操作控制寄存器的指令的监控来实现。首先设置 VMCS 中的 Primary Process-Based VM-Execution Controls 域的 active secondary controls 位（bit31）为 1，激活 Secondary Process-Based VM-Execution Controls 域，然后设置该域中的 Descriptor-table exiting 位（bit2）为 1。经过上述设置，当虚拟机中再执行 LGDT、LIDT、LLDT、LTR、SGDT、SIDT、SLDT 和 STR 这 8 条指令时，就会导致 VM exit。

对 LGDT 和 LIDT 两条指令陷入虚拟机监控器后采取的处理方式是直接跳过，具体实现方法为：首先计算当前陷入指令的长度，然后将客户机指令寄存器 RIP 中的值加上该指令长度得到一个新的值，再把这个新值写入到客户寄存器 RIP 中，使客户机从这个新指令地址处继续执行，从而实现指令的跳过。

对于其他 6 条指令，为了不影响它们的正常功能，保证系统的正常运行，根据这些指令功能的描述，在虚拟机监控器中增加对这些指令进行模拟处理的模块。当它们引起虚拟机陷入时，虚拟机监控器模拟执行这些指令的执行，然后返回虚拟机，从而保证了它们的执行结果不受监控而产生影响。

4.3.4　逻辑完整性检测

系统数据的逻辑完整性通过关联数据对象视图交叉对比来实现，实现的关键是基于虚拟机的视图构建，因为在虚拟监控器中没有操作系统语义信息，所以采用虚拟机自省技术，根据监控信息重新构建。

1. 系统底层视图构建方法

底层视图是通过 VMM 监控器监控进程的切换过程来绘制的。常用的方法是通过在 VMM 中监控 CR3 寄存器的值是否改变来识别进程切换事件，并以 CR3 作为进程的标识，如果发现新的 CR3 值，则将其作为进程标识加入进程列表，并同步识别进程退出事件以维护进程列表的最新状态[8]。这种方法存在的问题是：(1)可能存在遗漏进程，因为 Linux 为了提高任务切换的效率，在进程加载执行时，如果新进程的基地址和上一个进程基地址相同，则 CR3 不做更新；(2)当检测

到 CR3 切换时获取进程信息，此时进程切换过程还没有完成对 thread_info 内容的更新，获取的进程信息仍然是旧进程的，而该进程马上就要转为非执行状态，之后虚拟机中发生的各种事件其实与该进程无关，不利于对进程进行进一步的行为监控。因此，为了更准确地对进程切换进行监控，通过对 Linux 进程调度代码的分析，本文选取 CLTS 指令作为进程切换的监控对象。

CLTS 指令主要用于清除 CR0 第三位中的内容，即任务切换标志(Task Switched，TS)，每次任务切换，处理器都会设置该标志，并在解释浮点算术指令时检查该标志。Intel VT 支持对 CR0 各种写操作监控，当 CLTS 指令执行时产生 VM exit，VMM 获取进程信息，构建底层进程视图，从而保证该视图包含所有实际参与 CPU 调度的进程信息。

2. 用户层进程视图构建方法

用户层进程视图是用户能够得到的最直接的进程视图，某个进程是否存在于该视图中是判断该进程是否是隐藏进程的主要依据，也是判断进程隐藏成功与否的主要标准。无论采用何种方式隐藏进程都会体现在这个结果上。用户层进程视图可以通过使用应用层软件或系统工具软件(如 ps、top 等)获得的当前系统中所有进程的信息来建立。

5 实验结果及分析

基于上述思想和技术，实现了原型系统 IBTPS，从功能和性能两个方面对基于完整性检查的终端安全防护方法进行了测评。功能测试主要用于验证框架是否能监控、检测和发现对系统完整性的破坏行为，保护系统的完整性；性能测试主要测试系统开销及对客户机系统性能的影响。本文实验的硬件环境为 Intel core i5 650M 双核 CPU(含 Intel VT 支持)；CPU 频率为 3.4GHz；三级 4M cache；内存为 4G，监控的终端操作系统为 Ubuntu12.04(precise)32 位，内核为 linux3.2.0-29-generic。

5.1 功能测试

功能测试选取了 11 个典型 linux 内核级 rootkit 来验证防护框架的有效性。表 1 给出了各个 rootkit 的攻击目标(即影响的内核完整性相关对象)和实验结果。

表 1 实验结果

名称	攻击目标	实验结果
enyelkm	内核代码	不能加载
Mood-nt	内核代码 系统调用表	不能加载
Phantasmagoria	内核代码	不能加载
SucKIT	内核代码	不能加载
Superkit	内核代码 系统调用表	不能加载
all-root	系统调用表	不能加载
Rkit	系统调用表	不能加载
override	系统调用表	不能加载
adore-ng	系统调用函数 隐藏进程	不能加载 发现隐藏对象
phide	系统调用表 隐藏进程	不能加载 发现隐藏对象
knark	系统调用表 隐藏进程	不能加载 发现隐藏对象

实验结果表明，安全防护技术能够对终端安全防护进行实时完整性监控，能有效阻止对操作系统内核完整性相关对象的恶意篡改行为，并能检测发现系统内的隐藏对象。

5.2 性能测试

性能测试选取 lmbench3.0 作为测试集，通过对比测试程序在 5 种不同环境下的运行效率，检测引入本文防护技术对终端运行性能的影响。5 种测试环境分别为：终端系统直接运行于硬件平台(Phys)、Xen4.1.0(Xen)、qemu-kvm1.0(KVM)、BitVisor1.3(BitVsior)和 LHVisor(LHVisor)之上，在测试过程中 LHVisor 所有监控项全部开启，而其余虚拟机环境均未采取任何安全监控机制。其中，Xen 和 KVM 采取全虚拟化，Host OS 和 Guest OS 均为 Ubuntu12.04，各分配 2G 内存。测试结果如图 5 所示。

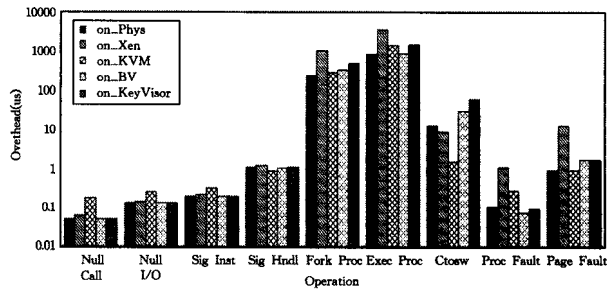

图 5 性能测试结果

从测试结果可知，本文的防护技术对系统性能造成影响不大，其原因主要是本文的 LHVisor 监控器采用半穿透方式，并未对系统调用、I/O 操作及信号处理进行陷入和监控，这些操作均未引入额外开销；而 Xen 和 KVM 由于采用全虚拟化方式，在这些操作上开销均有增加。由于 LHVisor 增加进程调度监控和处理，使任务切换相关开销有所增加(fork 进程后直接退出(Fork Proc)、执行 execve 调用再退出(Exec Proc)以及进程上下文切换(Ctxsw))，但开销仍低于 Xen 和 KVM。

结束语 本文从终端计算机安全防护的角度出发，提出了一种基于完整性检查的终端安全防护方法，将完整性监控和度量技术相结合，保证了终端操作系统内核的全时空安全。以 TPM 为硬件可信基，采用完整性度量方法建立从硬件平台到操作系统的基础可信链；在系统运行过程中监控内核代码、内核结构、寄存器和内核状态信息等完整性相关对象，发现和阻止对系统的恶意篡改行为，保证系统的完整性。利用 Intel VT 硬件辅助虚拟化机制采用半穿透技术设计实现了一个轻量级虚拟监控器，构建了防护原型系统，测试表明，该原型系统就终端实施有效保护时对其性能的影响较小。

参 考 文 献

[1] 沈昌祥，张焕国，冯登国，等. 信息安全综述[J]. 中国科学，2007，37(2)：129-150
Shen Chang-xiang, Zhang Huan-guo, Feng Deng-guo. Information Security Review[J]. Chinese Science, 2007, 37(2): 129-150

[2] Bratus S, D'Cunha N, Sparks E, et al. TOCTOU, traps, and trusted computing[M] // Trusted Computing-Challenges and Applications. Springer Berlin Heidelberg, 2008: 14-32

[3] 石晶翔，陈蜀宇，黄汉辉. 基于 Linux 系统调用的内核级 Rootkit 技术研究[J]. 计算机技术与发展，2010，20(4)：175-178
Shi Jing-xiang, Chen Shu-yu, Huang Han-hui. Research on Kernel Level Rootkit Technology Based on Linux System Call [J]. Computer Technology and Development, 2010, 20(4): 175-178

(下转第 174 页)

假设 B 的分布是一个 BDH 元组，即 $T=e(g,g)^{abc}$，则密文形式为：

$$C=(e(g,g)^{abc}M_d,\prod_{k=1}^{i}F_k(v_k)^{c+\rho},g_1^{\rho},g^c)$$

$$=(e(g_1,g_2)^cM_d,\prod_{k=1}^{i}F_k(v_k)^{c+\rho},g_1^{\rho},g^c)$$

显然，C 是明文 M_d 的一个有效加密密文；否则，T 是 G 中的一个随机元素。这种情况下，从密文得不到 B 所选择的比特 d 的任何信息。

阶段 2：A 继续发起私钥提取询问，B 的回复同阶段 1。

猜测：最终，敌手 A 输出两个猜测 $b',d'\in\{0,1\}$。如果 $b=b',d=d'$，B 输出 1，否则输出 0。若分布 $T=(g,g)^{abc}$，则敌手 A 满足 $|\Pr[b'=b\wedge d'=d]-\frac{1}{4}|>\varepsilon$。若分布 $T\neq(g,g)^{abc}$，则 $\Pr[b'=b\wedge d'=d]=\frac{1}{4}$。因此有：

$$|\Pr[B(g,g^a,g^b,g^c,e(g,g)^{abc})=0]-\Pr[B(g,g^a,g^b,g^c,T=0)]|\geqslant|(\frac{1}{4}\pm\varepsilon)-\frac{1}{4}|=\varepsilon$$

以上是对定理 1 的证明。

4 方案对比

本节对上述方案和一些经典 IBE 方案进行对比。对比结果如表 2 所列。

表 2 方案对比

方案	复杂性假设	安全级别	匿名性
Waters[7]	BDH	CPA	no
Boneh-Boyen[13]	BDHI	sID-CPA	no
Gentry[11]	ABDHE	CPA	yes
本文	BDH	CPA	yes

结束语 自 2005 年 Abdalla 提出在不依赖随机预言机模型下如何构造匿名 IBE 这一公开问题以来，匿名 IBE 就成为密码学的一个研究热点。目前，匿名 IBE 方案并不是很多。本文受 Gentry[11] 启发，基于 DBDH 问题构建了一个标准模型下的匿名 IBE 方案，并对方案作了正确性证明、匿名性分析以及安全性证明。与文献[11]相比，本文方案在加密阶段更有优势。值得一提的是，本方案弥补了 DBDH 问题下匿名 IBE 的空缺且达到了 CPA 安全。在今后通信技术高速发展的时代，必然对 IBE 的设计提出更高的要求。如何在标准模型下构造安全性和效率都较高的 CCA 安全的方案是一个值得深入研究的问题。

参 考 文 献

[1] Shamir A. Identity-based Cryptosystems and Signature Schemes [C]// Wagner D, ed. Advances in Cryptology-Crypto'84, Lecture Notes in Computer Science. vol. 196, Berlin: Springer-Verlag, 1984: 47-53

[2] Boneh D, Franklin M. Identity-based encryption from the Weil pairing[C]// Advances in Cryptology—CRYPTO 2001. Springer Berlin Heidelberg, 2001: 213-229

[3] Chatterjee S, Sarkar P. Identity-based encryption[M]. Springer, 2011

[4] Canetti R, Goldreich O, Halevi S. The random oracle methodology, revisited[J]. Journal of the ACM(JACM), 2004, 51(4): 557-594

[5] Boneh D, Di Crescenzo G, Ostrovsky R, et al. Public key encryption with keyword search[M]// Advances in Cryptology-Eurocrypt 2004. Springer Berlin Heidelberg, 2004: 506-522

[6] Abdalla M, Bellare M, Catalano D, et al. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions[C]// Advances in Cryptology-CRYPTO 2005. Springer Berlin Heidelberg, 2005: 205-222

[7] Waters B. Efficient identity-based encryption without random oracles[M]// Advances in Cryptology-EUROCRYPT 2005. Springer Berlin Heidelberg, 2005: 114-127

[8] Boneh D, Boyen X. Secure identity based encryption without random oracles[M]// Advances in Cryptology-Crypto 2004. Springer Berlin Heidelberg, 2004: 443-459

[9] Boneh D, Boyen X. Efficient selective identity-based encryption without random oracles[J]. Journal of Cryptology, 2011, 24(4): 659-693

[10] Boyen X, Waters B. Anonymous hierarchical identity-based encryption(without random oracles)[M]// Advances in Cryptology-CRYPTO 2006. Springer Berlin Heidelberg, 2006: 290-307

[11] Gentry C. Practical identity-based encryption without random oracles[M]// Advances in Cryptology-EUROCRYPT 2006. Springer Berlin Heidelberg, 2006: 445-464

[12] 胡亮，刘哲理，孙涛，等. 基于身份密码学的安全性研究综述[J]. 计算机研究与发展，2009，46(9)：1537-1548
Hu Liang, Liu Zhe-li, Sun Tao, et al. Survey of Security on Identity-Based Cryptography[J]. Journal of Computer Research and Development, 2009, 46(9): 1537-1548

[13] Boneh D, Boyen X. Efficient selective-ID secure identity-based encryption without random oracles[M]// Advances in Cryptology-EUROCRYPT 2004. Springer Berlin Heidelberg, 2004: 223-238

(上接第 166 页)

[4] Petroni N L, Hicks M. Automated detection of persistent kernel control-flow attacks[C]// Proc. of the 14th ACM Conference on Computer and Communications Security. New York: ACM Press, 2007: 103-115

[5] Baliga A, Ganapathy V, Iftode L. Detecting kernel-level rootkits using data structure invariants[J]. IEEE Transactions on Dependable and Secure Computing, 2011, 8(5): 670-684

[6] Trusted Computer Group. TCG Specification Architecture Overview, version1. 2[EB/OL]. https://www.trustedcomputinggruop.org

[7] Intel® 6 4 and IA-3 2 Architectures Software Developer's Manual Volume 3B: System Programming Guide[R]. Intel Corporation, 1997-2009

[8] 李博，沃天宇，胡春明，等. 基于 VMM 的操作系统隐藏对象关联检测技术[J]. 软件学报，2013，24(2)：405-420
Li Bo, Wo Tian-yu, Hu Chun-ming, et al. Hidden OS Objects Correlated Detection Technology Based on VMM [J]. Journal of Software, 2013, 24(2): 405-420

[9] Hofmann O S, Dunn A M, Kim S, et al. Ensuring operating system kernel integrity with OSck[J]. ACM SIGPLAN Notices. ACM, 2011, 46(3): 279-290

[10] Gadaleta F, Nikiforakis N, Mühlberg J T, et al. Hyperforce: Hypervisor-enforced execution of security-critical code[M]// Information Security and Privacy Research, Springer Berlin Heidelberg, 2012: 126-137