基于可信计算的动态完整性度量架构

刘孜文[①②] 冯登国[②]
[①](中国科学技术大学电子工程与信息科学系 合肥 230027)
[②](信息安全国家重点实验室 中国科学院软件研究所 北京 100190)

摘 要：该文提出一种基于可信计算的操作系统动态度量架构(DIMA)，帮助管理员动态地检查系统中进程和模块的完整性。相对于以往的各种操作系统度量架构，该架构能按需对系统中活动的进程或模块进行动态实时的完整性度量与监控，基本解决了其他架构难以避免的 TOC-TOU 问题，特别是针对某些直接对运行中的进程的攻击有很好的效果。另外，DIMA 实现了对对象细粒度度量——由度量整个文件实体细分为度量代码、参数、堆栈等等。最后给出了基于 Linux 操作系统的动态度量原型实现，在实现中使用了基于可信平台模块(TPM)作为架构的信任源点，测试结果表明 DIMA 能够实现预定目标且有良好的性能。

关键词：可信计算模块；完整性度量；动态完整性度量架构(DIMA)
中图分类号：TP309 **文献标识码**： A **文章编号**：1009-5896(2010)04-0875-05
DOI: 10.3724/SP.J.1146.2009.00408

TPM-Based Dynamic Integrity Measurement Architecture

Liu Zi-wen[①②] Feng Deng-guo[②]
[①](*Department of Electronic Engineering and Information Science*, *University of Science and Technology of China*, *Hefei* 230027, *China*)
[②](*State Key Laboratory of Information Security*, *Institute of Software*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

Abstract: This paper presents a TPM-based architecture DIMA (Dynamic Integrity Measurement Architecture), which helps the administrators check the integrity of the processes and modules dynamically. Compares with other measurement architectures, DIMA uses a new mechanism to provide dynamic measurement of the running processes and kernel modules. Some attacks to running processes which use to be invisible to other integrity measurement architectures can be now detected. In this case, DIMA solves the TOC-TOU problem which always bothers others before. In addition, instead of measuring the whole file on the hard disk, the object is divided into some small pieces: code, parameter, stack and so on to make a fine-grained measurement result. Finally, the DIMA implementation using Trust Computing Module (TPM) is discussed and the performance data is presented.

Key words: Trusted Computing Module (TPM); Integrity measurement; Dynamic Integrity Measurement Architecture (DIMA)

1 引言

随着计算机应用的普及，对计算机系统的完整性保护日益受到重视，出现了各种完整性保护的模型和实施方法。度量方法，是一个较新的方法。它的原理是在某些特定的时刻，对目标进行度量，得到目标的某些信息(比如对文件的散列值)，将这些信息的值与事先记录的标准值进行比较，从而判断目标的完整性是否被破坏。度量方法不仅其本身是一种保护完整性的安全机制，还为其他的安全机制如访问控制，远程证明等提供基本的安全数据。在可信计算的概念提出和推广后，度量作为可信计算的核心内容之一，其重要性得到了进一步提高。

度量方法包括几个关键要素：度量的时间点，度量的对象，度量在系统中实现的层次。其中度量的时间点决定了在什么时刻进行度量；度量的对象决定了它对整个计算机系统的保护轮廓；度量的实现层次决定了度量系统本身的安全性和实现的复杂性。

基于度量点设置的不同，度量系统可以分为静态度量，周期性度量，实时动态度量，度量点的密度和分布是衡量一个度量系统度量能力的主要标准之一。在传统的度量架构中，度量点都设置在某个

2009-03-26 收到，2009-10-09 改回
国家 863 计划项目(2007AA01Z412)和国家科技支撑计划项目(2008BAH22B06)资助课题
通信作者：刘孜文 sophyhare@163.com

或某些事先规定好的时刻(也就是静态度量或者周期性的度量)。这种度量点的设置有如下几个不足：(1)度量点由系统设计者决定，而不是由实际的使用者决定，因此在灵活性上受到影响；(2)遭遇典型的 TOU-TOC 问题：进程在被度量前未受攻击，而在度量后遭到破坏，静态度量就会给出错误的结果，周期性度量如果其度量周期长于进程完整性被损坏的时间，也可能发现不了这样的破坏。

度量的另一个要素——系统中实现的层次的问题，在以前的大部分架构中都没有很好地得到解决，直到可信计算技术的出现。1999 年 TCPA(Trusted Computing Platform Alliance，后改名为 TCG)[1]制定了可信计算的标准，主要表述了如何利用可信计算模块 TPM[1]提供的各种功能来保护系统平台的安全。TPM 本身提供了安全随机数产生器、非易失防篡改存储、密钥产生算法、RSA 加解密密码函数以及 hash 函数 SHA-1 等功能，可以看出，TPM 是一种提供了某些安全功能的硬件(固件)，同时它的功能和规格比较统一，所以是设计各种计算机安全系统的可信计算基 TCB(Trust Computing Base)的很好选择。度量方法通过将度量系统的信任根构建在基于 TPM 的硬件环境中，能获得比以前各种基于纯软件的系统更好的安全性。

DigSig[2]系统在可执行文件头部加入安全信息，并在运行时验证这些安全信息，来解决对 Linux 可执行程序非法篡改的问题。缺点是 DigSig 自身的安全没有得到保障。CoPilot[3]系统独立于系统 CPU 之外使用一个 PCI 板卡的协处理器进行度量工作，以确保度量系统本身的安全性。CoPilot 的缺点在于设计实现复杂，需要进行多次对内存的映射操作，而且只能先设定度量周期，不能真正实现动态的度量。这两个系统代表了可信计算发展早期，度量方法中纯软件的方法以及结合一些硬件的方法。第 1 种实现简单，但是度量系统以及度量结果本身的安全性容易遭到破坏。第 2 种需要对系统进行硬件的改造。近几年，很多计算机都加装了 TPM 芯片，从而使得度量系统能在不对计算机进行过多的改造的基础上，实现硬件级别的安全支持。

IMA[4]是 IBM 研究院实现的基于可信计算的度量系统，在操作系统将程序载入到内存中的时刻，对程序文件进行度量，度量结果交由可信计算芯片 TPM 进行处理。这样，完成一条从 TPM 硬件 —— 完整性度量系统 —— 应用程序的信任链，但由于是在系统调用中插入了度量点，因而会产生大量冗余。IMA 的后继工作 PRIMA[5]针对此进行了改进，和 SELinux 结合，使用了策略规约的方法减少冗余度量。IMA 和 PRIMA 从本质上说，仍然是静态度量系统。BIND[6]系统由程序员自己决定度量点并在度量点插入 BIND 提供的 hook 函数接口，从而提高了度量精度。然而它增加了程序员编写程序的负担，且与以前的所有程序都不兼。LKIM[7]以及后继的工作[8]针对系统内核进行度量，它在静态度量的基础上，定义一系列变量表示系统的状态，这些变量值发生变化的时候，重新进行度量，从而实现动态度量的目的，它的主要不足如下：首先它强调的静态度量加上状态变量不能表示真正的动态的度量；其次它度量针对的是 Linux 内核，对普通的进程无能为力。文献[9]主要关注的是程序执行结果的远程证明，通过度量在整个执行期间目标进程与别的进程、文件的数据交互，来对整个程序的执行过程进行证明。

以上这些现有的各种度量系统在动态性和实时性方面都有很大的缺陷，而动态攻击并非不存在，这方面可以由自修改代码 SMC 技术中得到一些启示：自修改代码 SMC[10](Self-Modifying Code)，是一类程序的统称，执行过程中会有意修改自身的代码，使得实际运行的代码和程序执行之前的代码的静态二进制表示不相同。自修改代码可以有效隐藏代码、数据以及程序的执行流程等，广泛应用于软件保护和恶意代码等领域。SMC 行为一般分为两步，首先修改存放代码的内存页的读写权限，然后对代码进行覆盖，以达到自己修改代码的目的。一种针对用户进程完整性的攻击也是基于类似原理，只是需要额外的步骤——操作系统中为了保护进程，一般都设定了不同进程的地址空间不能互相访问，所以需要在修改读写权限之前，恶意进程需要先冒充为目标进程。2.1 节中说明了这种攻击的原理，4.1 节实验将展示这种攻击。

针对度量系统的实时性问题，本文设计了动态完整性度量架构(Dynamic Integrity Measurement Architecture，简称 DIMA)。它的主要特点是对计算机系统中的进程进行动态的完整性度量，即在任意时刻，对正在运行的系统中的进程(以及模块)的有机构成进行完整性的度量，并在度量的过程中使用 TPM 保护度量架构和对度量结果进行签名以提高整个系统的安全性。本文第 2 节介绍 DIMA 的设计思路和架构，第 4 节描述 DIMA 的具体实现和工作步骤，第 5 节给出测试结果，第 6 节总结了全文。

2 架构设计

2.1 面临的主要问题

DIMA 主要需要解决的问题是：动态实时地发

现和报告针对系统中已经运行的进程的完整性的攻击。例如，对于代码修改的方法而言：首先恶意进程伪装成目标进程欺骗操作系统，使得自己能够获得对目标进程地址空间的内容进行修改的权限；接着对地址空间中存放代码的部分的页面属性进行修改；然后对这些存放代码的页面内容进行修改，完成恶意进程希望实现的功能；最后视情况将进程代码恢复为原来未被修改的模样。整个过程见图 1。

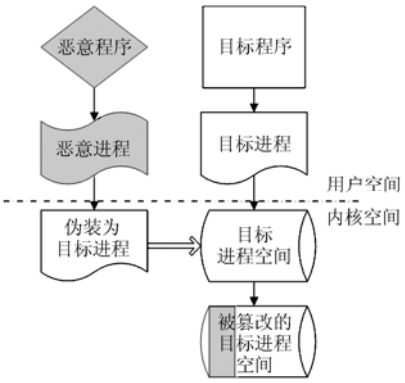

图 1 基于代码修改方法的攻击原理示意图

可以看出，这样的攻击有几个特点：一是入侵的时机是进程运行之后，因此各种载入时的度量系统如 IMA，PRIMA，DigSig 等就不能检查出来，对于周期检查的 CoPilot 系统，只要入侵完成在度量周期之内，度量系统也是无能为力。二是入侵的对象是普通的用户进程，这样 LKIM，BIND 系统就不能有效的做出反应。三是入侵修改的内容是程序的代码部分，因为程序的代码部分一旦被修改，整个程序的功能就会完全改变，从而实现入侵目的。

2.2 设计思路

根据度量架构的几个要素和对面临问题的分析，DIMA 架构的设计主要集中于 3 个方面的考虑：度量对象、度量点、度量架构在系统中的位置。

(1)关于度量对象：只有各种正在运行的进程(或者模块)的完整性才能表达系统当前时刻的完整性，因而 DIMA 的度量对象是当前系统中的进程。程序运行的时候，被划分为几个部分加载到内存中，为了实现更加细粒度的度量，DIMA 将针对内核的划分对进程的代码，参数堆栈等分别进行度量。

(2)关于度量点：早期的度量系统以及后来的很多改进版本都采用了单一度量点(最普遍的是载入时度量)。对于系统中一些守护进程，在开机时被加载直至下次开机才被重新加载的情况，这样的方法显然不合适。DIMA 采用的是动态度量的方法，即在任意需要的时刻，都能对度量对象进行度量。

(3)关于度量架构的位置：事实上所有的安全功能架构都面临自身安全的问题，因此有可信计算基 TCB(Trust Computing Base)的提出，只要 TCB 部分不被攻破，安全架构自身就是安全的。DIMA 围绕可信计算芯片 TPM 来设计 TCB，通过 TPM 对度量架构本身和度量结果做验证。

2.3 架构

为了同时满足这些要求，遵循设计思路，DIMA 将跨度于用户空间和内核空间，整个架构见图 2。

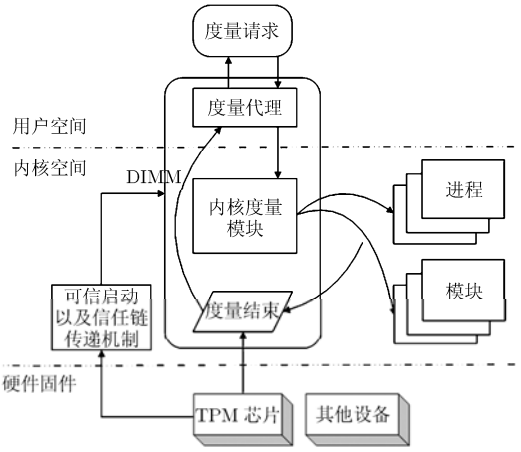

图 2 度量架构

度量代理 MA(Measurement Agent)处于用户空间，用户的度量请求通过它的接口提交给内核，后继工作中的远程证明等功能也是与它进行交互。内核度量模块 KMM(Kernel Measurement Module)，处于内核空间，完成整个动态度量的主体工作。可信启动以及信任链传递机制在事前的工作中完成[11]，作用是保证整个 DIMA 架构的安全，其中可信启动在系统启动时对 BOIS，boot loader，操作系统内核进行完整性度量，并将度量结果存储在 TPM 的 PCR 当中，保证了系统初始状态的可信；信任链传递机制对 DIMA 本身进行了度量，确保 TPM 的基于硬件的安全性能够延伸到 DIMA 上，具体的流程见参考文献[11]。

2.4 度量方法

(1)对系统中活动进程的度量　Linux 系统为每个进程维护一个映像，该映像完全反映了进程的功能和信息。对进程的动态度量，就是在任意接收到度量请求的时刻，对度量目标——某个进程在内存中的映像进行度量。这个映像都在内存中某个可寻址的位置。系统为了管理，为每个进程地址空间维护一个内存描述符(mm_struct)，用以记录与该进程地址空间相关的全部信息，位置见图 3。

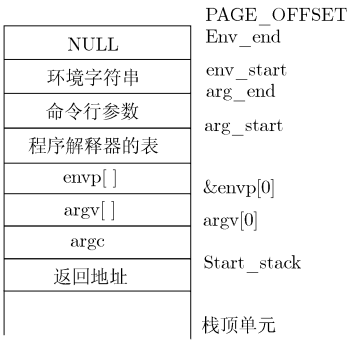

图 3 进程地址空间

对于某一个进程而言，其运行过程和运行结果可以进程地址空间内的数据决定。这些数据中，有几处数据最为关键。Start_code 和 end_code 标识程序代码所在的线性区的起始和中止地址；Start_data 和 end_data 标识程序数据所在的线性区地址；Arg_start 和 arg_end 标识命令行参数所在的堆栈部分地址。他们是度量的主体。

(2)对内核模块的度量 Linux 系统为每一个模块维护一个结构 struct module，内核其他部分要与模块进行数据交互可以通过访问代表这个模块的 module 结构进行；所有的模块都通过模块链表 modules 连在一起。DIMA 通过对内核模块中这些关键数据的度量，来实现对模块的度量。

3 度量流程

3.1 度量代理 MA(Measurement Agent)

(1)格式化接收到的度量请求(可能来自于远端程序的信任挑战，也可能来自于本地的安全需求)；

(2)把请求发给内核的度量模块；

(3)待内核空间的 KMM 处理完后，把接受到的度量信息组装，发送回远方或者本地。

3.2 内核模块 KMM(Kernel Measure Module)

(1)通信。我们使用的是 Linux 提供的/proc 虚拟文件系统机制：当 KMM 模块插入内核的时候，在虚拟文件系统/proc 中自动生成/proc/dynamic_measurement 文件。

(2)对进程进行度量。接受到度量代理发送的度量请求后，内核度量模块判定其是进程还是模块，如果是进程，步骤如下：

(a)解析请求中的进程名，并在内核维护的进程链表中找到这个进程，获得进程的句柄；

(b)通过进程的句柄找到进程的各种信息所在的地址，包括进程的代码段，数据段，参数段，堆栈段等等，此时，获得的是地址是相对于目标进程自身的线性地址；

(c)将它转换为物理地址并读取物理地址中内容，映射到 KMM 的进程空间中；

(d)将这些内容交由 TPM 芯片做散列值，并由 TPM 进行签名，将散列值和签名值存入事先定义好的数据结构中。

(3)对模块进行度量。接受到度量代理发送的度量请求后经判定为模块，进行如下操作：

(a)解析请求中的模块名，并在内核维护的模块链表中找到模块，获得模块句柄；

(b)通过句柄找到模块的正文的地址(*module_core)，长度(core_text_size)

(c)使用和度量进程一样的步骤进行度量。

(4)度量架构的安全启动。操作系统启动之前，由 TPM 对 BIOS 进行度量，启动 BIOS 后对 Grub 进行度量，最后对操作系统的映像进行度量，从而保证度量架构启动前的安全性，度量架构启动之后 TPM 对度量架构的关键部分如 KMM 模块和配置文件进行度量。具体的流程参见文献[11]。

4 测试

4.1 攻击测试

使用基于 SMC 原理的攻击方法，对活动的进程进行攻击，攻击进行中使用 DIMA 对此进程进行度量：在 Linux 下编译一个类似于 hello world 程序，在打印“hello world”之前等待，直到控制台得到键盘输入(模仿一个持续性的应用)。该程序等待时，使用基于 SMC 的方法修改代码段，这里只用全 0 覆盖，覆盖的长度按需要控制。使用 DIMA 对覆盖前和覆盖后的 hello world 程序进行度量。

结果：无论长度如何，DIMA 都能检测到入侵。覆盖的长度如果短，程序仍然正常运行结束，如果覆盖的长度过长，则程序不能按照正常步骤输出 hello world 字样。可以看出，基于 SMC 的攻击动态改变了程序的运行，但是 DIMA 能够很清楚的看到度量值发生了变化。而在以往的架构中，做不到这一点。作为对比使用了提供源码的 IMA[4]架构进行了测试，对该攻击，IMA 不能检测到。

4.2 效率测试

实验中对两个进程 Gaim 和 nm-applet 进行了度量，对两个模块 ata_piix 和 ext3 进行了度量，并对所有模块的集合做了度量，耗时如表 1 所示：

表 1 DIMA 性能测试表

	进程		模块		
	nm-applet	Gaim	ata_piix	Ext3	所有模块集合
度量耗时(ms)	3.7	12.4	0.77	2.3	94.7

由测试可以看出，对一个进程做度量所需要的时间为 10 ms 数量级(视进程的大小而定)，对一个模块做度量所需要的时间为 1 ms 数量级，对系统造成的负担很小。

5　结束语

本文介绍了一种基于可信计算的动态度量架构，并做了相应的实现。相对于现有的各种度量架构，它最大的特点是实现了实时的动态度量，解决了度量中最关键的 TOC-TOU 问题，并把度量对象扩展到进程，模块以及它们的周边信息，并利用可信技术芯片 TPM 进行硬件级别的保护。测试表明 DIMA 能够完成对一些实时攻击方法的检测，性能方面在对时间要求比较高的场合也有很好的表现。今后的工作包括几个方面，一是拓展对进程的度量，加入对进程其他安全信息的度量；二是考虑如何将此架构应用到访问控制之中，使得度量结果能得到更为广泛的应用。

参 考 文 献

[1] Trusted Computing Group. TCG PC client specific implementation specification for conventional bios version 1.2, July 2005.

[2] Apvrille A, Gordon D, Hallyn S, Pourzandi M, and Roy V. DigSig: Run-time authentication of binaries at kernel Level[C]. Proceedings of LISA '04 Eighteenth Systems Administration Conference. Atlanta, GA, USENIX Association November, 2004: 59–66.

[3] Petroni N Jr and Fraser T, *et al*.. Copilot —— A coprocessor -based kernel runtime integrity monitor[C]. Proceedings of the 13th conference on USENIX Security Symposium. San Diego, CA, 2004, Vol. 13: 13–13.

[4] Sailer R, Zhang Xiao-lan, Jaeger T, and Van Doorn L. Design and implementation of a TCG-based integrity measurement architecture[C]. Proceedings of USENIX Security Symposium. Lake Tahoe, California, USA, ACM Press, Aug. 2004: 223–238.

[5] Jaeger T, Sailer R, and Shankar U. PRIMA: Policy-reduced integrity measurement architecture[C]. Proceedings of the eleventh ACM symposium on Access control models and technologies. Lake Tahoe, California, USA, 2006: 19–28.

[6] Shi E, Perrig A, and Van Doorn L. BIND: A fine-grained attestation service for secure distributed systems[C]. Proceeding of the IEEE Symposium on Security and Privacy. Oakland, CA, USA, IEEE Press, 2005: 154–168.

[7] Loscocco P A, Wilson P W, Pendergrass J A, and McDonell C D. Linux kernel integrity measurement using contextual inspection[C]. Proceedings of the 2007 ACM workshop on Scalable trusted computing. Alexandria, Virginia, USA, 2007: 21–29.

[8] Thober M and Pendergrass J A. McDonell C D: Improving coherency of runtime integrity measurement[C]. Conference on Computer and Communications Security Proceedings of the 3rd ACM workshop on Scalable trusted computing. Alexandria, Virginia, USA, 2008: 51–60.

[9] Gu Liang, Ding Xu-hua, Deng R H, Xie Bing, and Mei Hong. Remote attestation on program execution[C]. Conference on Computer and Communications Security Proceedings of the 3rd ACM workshop on Scalable trusted computing. Alexandria, Virginia, USA, 2008: 11–20.

[10] Wu Yong-dong, Zhao Zhi-gang, and Chui Tian-wei. An attack on SMC-based software protection[M]. Springer Berlin / Heidelberg. 2007: 232–248.

[11] 徐震，沈丽红，汪丹. 一种可配置的可信引导系统. 中国科学院研究生院学报, 2008, 25(5): 626–630.
Xu Zhen, Shen Li-hong, and Wang Dan. LOIS grub: A configurable trusted booting system[J]. *Journal of the Graduate School of the Chinese Academy of Science*, 2008, 25(5): 626–630.

刘孜文：　男，1981 年生，博士生，研究领域为操作系统安全、可信计算.

冯登国：　男，1965 年生，博士，研究员，博士生导师，主要研究领域为网络与系统安全.