一种基于安全优先架构的细粒度可信监测度量方法

田 竞[1,2], 孙慧琪[1], 武希耀[1], 贾晓启[1,2], 张伟娟[1,2], 黄庆佳[1]

[1]中国科学院信息工程研究所 北京 中国 100049
[2]中国科学院大学网络空间安全学院 北京 中国 100049

摘要 Linux 下的 Rootkit 通常使用修改系统内核关键位置数据的手段破坏系统内核完整性。可信计算是保护系统内核完整性的重要方法，可以使用它对 Rootkit 攻击进行监测。相较传统的被动可信计算体系，主动可信计算体系因其对上层应用透明、安全机制与计算功能充分隔离、可信根完全受硬件保护等特点，可以更有效地进行系统内核完整性保护。但目前的主动可信监测度量方法存在监测结果粒度较粗的问题，不能为防御者进行攻击对抗提供更详细的信息。针对这一问题，本文提出了一种基于安全优先架构的细粒度可信监测度量方法，安全域通过解析计算域内存语义信息，实现符号级别的细粒度可信度量，得到可用来对攻击进行分析的监测结果。实验表明，该方法可以在计算域受到 Rootkit 攻击时检测到全部被篡改的.text 和.rodata 段的符号，使用该方法得到的细粒度监测结果可以用来分析 Rootkit 的攻击手段和攻击目的，同时该方法对计算域的性能几乎没有影响。

关键词 可信计算；安全优先架构
中图法分类号 TP309.2 DOI 号 10.19363/J.cnki.cn10-1380/tn.2019.09.05

A Fine-grained Trusted Monitoring Measurement Method Based on Security-first Architecture

TIAN Jing[1,2], SUN Huiqi[1], WU Xiyao[1], JIA Xiaoqi[1,2], ZHANG Weijuan[1,2], HUANG Qingjia[1]

[1] Institute of Information Engineering, Chinese Academy of Science, Beijing 100049, China
[2] School of Cyber Security,Unneversity of Chinese Academy of Science, Beijing 100049, China

Abstract Rootkit under Linux usually destroys the integrity of the system by modifying the key location data of the system kernel. Trusted Computing is one of the important methods to protect system integrity, which can be used to monitor Rootkit attacks. Comparing with the traditional passive trusted computing system, the active trusted computing system is transparent to the upper application, its security mechanism and computing function are fully isolated, and the trusted root is completely protected by hardware. So it can protect the integrity of the system kernel more effectively. However, the current active trusted monitoring measurement methods have the problem of coarse granularity of monitoring results, which can not provide more detailed information for defenders to carry out attack confrontation. To solve this problem, this paper proposes a fine-grained trusted monitoring measurement method based on security-first architecture, Security domain implements fine-grained trusted measurement at symbol level by parsing the memory semantic information of computation domain, and obtains the monitoring results that can be used to analyze attacks. Experiments show that this method can detect all tampered symbols of text and rodata segments when the computing domain is attacked by Rootkit. Fine-grained monitoring results obtained by this method can be used to analyze Rootkit's attack means and purpose, and it has little effect on the performance of the computing domain.

Key words trusted computing; security-first architecture

1 引言

Linux 下的 Rootkit 通常使用修改系统内核关键位置的手段破坏系统内核完整性，进而在攻击目标上隐藏指定的文件、进程和网络链接等信息。系统内核完整性保护是安全领域的重点研究方向之一，

通讯作者：张伟娟，博士，助理研究员，Email: zhangweijuan@iie.ac.cn。
本课题得到中国科学院网络测评技术重点实验室资助项目，网络安全防护技术北京市重点实验室资助项目，北京市科技计划课题（No.Z191100007119010），国家自然科学基金(No.61772078)资助。
收稿日期: 2019-06-04；修改日期: 2019-08-15；定稿日期: 2019-08-20

是进行 Rootkit 防御的重要手段。系统内核完整性保护可大致分为普通的软件方法、基于虚拟化的方法、普通的硬件方法、可信计算方法四种类型。与操作系统处于同一特权级别的软件系统内核完整性保护方法实现方便，但理论上与操作系统内核在相同特权级别上运行的恶意软件可以绕过任何运行于该特权级别的安全防护措施。基于虚拟化的系统内核完整性保护的方法可以让安全机制的特权级别高于操作系统，但虚拟机管理程序本身可能成为攻击目标，且在虚拟化环境中运行操作系统还会产生高达 30%的性能开销，所以该方法在某些看重运行性能的情况下并不适用[1]。使用硬件外围设备的系统内核完整性保护方法需要大量的新硬件,而且这些硬件都受主机端 CPU 控制，在 CPU 启动之后才能启动，可能会被主机端的攻击禁用。关于可信计算如何应用于系统内核完整性保护的研究一直在进行[2-3]，但传统的被动可信防御方法与其他利用软件和硬件进行系统内核完整性保护的方法存在类似的问题，其可信平台模块(TPM, Trusted Platform Module)需要调用才能提供可信计算功能，而调用程序的特权级别低于或等于操作系统内核，可以被攻击者影响。

安全优先架构[4]是近期提出的一种用来进行系统内核完整性保护的新型主动可信防御体系架构，它分为隔离的安全域和传统的计算域两个部分。相较普通的硬件完整性保护方法，在安全域中，该体系架构采用独立的安全处理器，且其特权级别高于计算域，启动先于计算域，运行不受计算域控制，不会被主机端的攻击行为影响。相较普通的软件完整性保护方法，安全优先架构中的安全机制在安全域中运行，与计算域隔离且特权级高于计算域中的操作系统，所以计算域中的攻击无法绕过安全机制。相较基于虚拟化的完整性保护方法，安全优先架构中的安全机制运行于安全处理器，与正常的计算工作分离，大大降低了安全机制的运行对计算域性能的影响。相较传统的被动可信防御方法，主动可信防御体系架构中的安全机制自主运行，不需操作系统和上层应用进行调用。

然而，在现有的基于安全优先架构的可信度量机制中，与计算域隔离的安全处理器缺乏对计算域内存中存储的二进制数据语义信息的获取，进而无法对计算域中的待度量区域进行进一步的语义划分。因此，现有的主动可信监测度量机制只能监测到攻击的发生，无法有效的对攻击进行更细粒度的定位与分析。

为有效支持对攻击进行归纳分类等细粒度分析，设计并实现一种细粒度的可信监测度量方法显得尤为重要。首先，知道攻击者对哪个符号进行了修改，就可以有效推测出攻击者的行为与目的，进而快速进行漏洞的定位和修复工作。如图 1 所示，若发现内核中的函数 tcp4_seq_show 被攻击者篡改，则可分析攻击者的目标可能是对某些 tcp 连接进行隐藏。但粗粒度度量只能监测到攻击发生在.text 段，无法据此推测攻击者的目的。其次，由于不同的 Rootkit 采用的攻击手段不同，使得精确到符号级别的细粒度度量的监测结果存在差异性，因此可根据细粒度的监测结果对不同的攻击行为进行分类，分析攻击者使用的攻击手段。在该攻击属于未知的新型攻击时，细粒度的度量方法能够分析出与未知攻击类型最相近的 Rootkit。对于采用不同攻击方式的 Rootkit，针对它们进行粗粒度度量获得监测结果很可能相同，不存在差异性，即无法达到细粒度度量的效果。

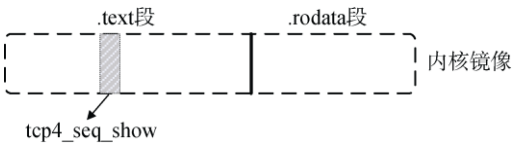

图 1　Rootkit 攻击示例

Figure 1　Rootkit attack example

综上所述，仅实现粗粒度的主动可信监测度量是不够的，为了给攻击对抗工作提供更多有价值的信息，对当前的基于安全优先架构的主动可信监测度量的效果进行改进，本文提出了一种基于安全优先架构的细粒度主动可信监测度量方法。通过符号信息的可靠传递，本文克服了硬件隔离导致的安全域获取计算域语义信息的困难，在安全域中实现了对计算域内存的符号级别的细粒度度量，提升了基于安全优先架构的主动可信监测度量结果的精度。实验表明，本文提出的细粒度主动可信监测度量方法可以成功实现对计算域内存进行符号级别的监测，同时对计算域性能的影响较小。

2　背景与相关工作

可信计算是实现系统内核完整性保护的方法之一，它分为被动可信和主动可信两种防御方式。除了可信计算以外，还有其他的基于硬件的防御技术提供了对系统内核完整性的保护。本节将分别对被动可信防御、主动可信防御、其他基于硬件的系统防御这三个部分进行介绍。

2.1　被动可信防御

可信平台模块(TPM, Trusted Platform Module)是

现在国外使用较多的被动可信计算技术。TPM 芯片提供加密等可信计算相关的功能，安全应用程序通过 TCG 软件协议栈(TSS, TCG Software Stack)来使用其提供的安全功能。基于可信计算组织(TCG, trusted computing group)提出的 TPM1.2[5]和 TPM2.0[6]规范，许多厂商已经实现了硬件 TPM 芯片，并且在一些计算机上成功应用。

目前对于 TPM 的研究主要集中于两方面。一是对于其编程接口的研究。由于 TPM 硬件使用时需要上层的调用，所以其编程接口成为了研究重点。Chen Liqun 等人[7]回顾了所有 TPM 2.0 ECC 功能，并讨论了现有的 TPM 命令是否可用于实现新的加密算法，证明了 ISO/IEC 18033-2 中指定的四种非对称加密方案可以使用 TPM 2.0 芯片实现。Jan Camenisch 等人[8]提供了 TPM2.0 接口的更好的规范，在对当前 TPM2.0 命令进行很少修改的前提下，解决最新的 TPM2.0 规范中直接匿名认证的严重缺陷。二是 TPM 芯片在实际安全防护中的应用。Penglin Yang 等人[2]将 CFI 和动态可信计算相结合，将运行时可信验证器 (RTTV) 作为一种基于 TPM 的动态 CFI 测量工具。Wang Yong 等人[3]改进了 linux 内核完整性度量体系结构(IMA)，并设计了基于 TPM2.0 的内核完整性度量框架。同时在 TPM2.0 芯片的基础上，实现了支持 TPM2.0 规范的 Linux 可信内核。

可信计算的可信根是整个可信链的基础，通常依靠硬件保护来确保可信根的可信。但 TPM+TSS 的体系结构中，可信根除了硬件以外还包含 BIOS 中的一部分代码，无法完全由硬件保护。此外，相比本文使用的安全优先架构，被动可信的体系一旦上层运行的不是经过修改的、加入了 TPM 调用接口的程序，或者上层程序被攻击者破坏，可信硬件就无法发挥作用。

2.2 主动可信防御

主动可信的概念是由国内学者提出的[9]，针对的是被动可信防御技术存在的对上层应用不透明、可信计算基未完全受到硬件保护等问题。

国内与其相关的研究较多。黄坚会等人[10]基于之前的研究[9]提出三阶三路(3P3C, three phases three channels)计算机架构防护理论，将之前的可信平台控制模块(TPCM, trusted platform control module) 完善为基于双系统体系结构的可信计算。但通过主机 CPU 启动可信度量的方式使得可信度量功能可能会受到主机端状态的影响。

为了防止存储在终端的敏感数据泄露，针对已有数据泄露防护技术的不足，余洋等人[11]提出一种基于 TPCM 的数据泄露防护模型。将可信平台控制模块与数据密封技术相结合，将被加密数据与平台状态绑定。通过可信平台控制模块进行密钥存储和管理，可以防止密钥遭受恶意病毒或者木马的破坏或盗取，还可以防止内部用户泄露系统中的重要数据。但这种方法是针对文件的防护，不是对运行时内存的防护，而且没有进行实验验证模型的效果。

此外还有使用独立安全处理器的主动可信计算模型[12]，在该模型中，主动可信度量由与常规处理器并行运行的安全处理器发起，安全处理器在物理隔离的基础上很难被攻击，可以有效保证整个系统的安全性。但由于物理隔离让安全处理器对主机内存语义信息的获取非常困难，目前用这些方式实现的主动可信度量的结果粒度较粗，度量效果有待提升。

2.3 其他基于硬件的系统防御

copilot [13]是一种基于协处理器的内核完整性监视器，用来监视主机内核是否被恶意修改。但相比安全优先架构中的安全处理器，这种协处理器可能会被主机端禁用或被主机端的攻击影响而不能正确访问物理地址，而且在系统启动后才能上电，没有对启动过程的可信检查。Liu Z.等人[14]利用 DRAM DIMM 检查系统的完整性。它可以监视所有内存流量并检测系统完整性违规。这种方法是利用内存流量信息判断内存关键位置是否被篡改的，只关心系统调用表、ITD 和许多其他内核函数指针等感兴趣的数据结构，在这范围之外的恶意篡改无法判断。Moon H. 等人[15]使用单独的硬件来监测总线流量以确保内核的完整性。所有处理器指令和 I/O 设备、内存和处理器之间的数据传输都必须通过系统的总线，通过监视此关键路径，该方法可以观察所有活动以查找恶意系统事务。这种方法可以捕获瞬态攻击，即在成功攻击系统后可以隐藏自身存在的攻击。但文章只在小型嵌入式系统中进行了实验，在其他平台上运行需要根据平台特点和限制进行修改。基于 SMM 的方法[16-19]使用 Intel SMM[20]来确保系统的可靠性，其可信计算基由 BIOS 和 SMM 组成，不完全受硬件保护。

3 威胁模型

Rootkit 常常采用修改内存关键数据的方式在正常的运行过程中隐蔽地执行攻击者注入的恶意代码，这使得对内存关键数据的可信度量在监测此类攻击时非常有效。可信度量将某时刻计算出的度量值与可信度量的基准值进行对比，根据比较结果确定被

度量的内容是否被篡改。

基于安全优先架构[4]的主动可信度量是目前比较先进的主动可信体系结构。这一架构使用与传统计算域CPU硬件隔离的安全域CPU运行安全相关的任务，对传统的计算域进行监测，保证了安全机制的可信性不受计算域的影响。但其采用的硬件隔离方式也带来了一些应用上的难题。硬件隔离导致安全域难以获取计算域内存二进制数据细粒度的语义信息，所以目前基于该体系结构只实现了粗粒度的可信监测度量。

Linux 系统的 Rootkit 经常通过对 Linux 内核内存空间进行篡改来实现攻击，例如 hook 系统调用表和修改函数序言将执行转移到替换程序。系统调用表位于 Linux 内核.rodata 段，内核函数的序言位于 Linux 内核的.text 段。AFkit 和 Suterusu 是使用这两种方式进行攻击的典型案例，其使用的攻击方式被目前的大多数 Rootkit 采用。

因此，本文只关注通过对计算域内核的.text 和.rodata 段进行修改来实现攻击目的的 Rootkit，不考虑修改驱动模块和用户进程的攻击行为。在安全优先架构的基础上，我们设计可靠的语义信息传递方案，向安全域传递计算域内核.text 和.rodata 段的所有符号级别的语义信息，在安全域实现对这两者的符号级别的细粒度可信监测度量。

4 系统设计

4.1 设计思路

要在安全域中实现对计算域内核的细粒度度量，必须给安全域提供足够的细粒度信息来进行计算域内存二进制数据语义信息的重构。对于细粒度度量来说，只有明确了待度量的二进制信息每个区域代表的意义，才能将某块内存区域的数据进行有意义的划分，进而对其实现细粒度度量。但在安全优先架构中，安全域和计算域是物理隔离的两个并行运行的系统，在安全域中访问计算域内存只能得到无具体语义的二进制信息。因此为了能在安全域中实现度量精度达到符号级别的细粒度度量，需要设法向安全域传递计算域的符号信息，进而在安全域内对计算域内存的语义信息实现重构。

为了在安全域完成计算域内存的语义重构，进而实现细粒度度量，我们将计算机的运行过程分为三个阶段：离线预处理阶段、启动阶段、运行阶段，如图 2 所示。

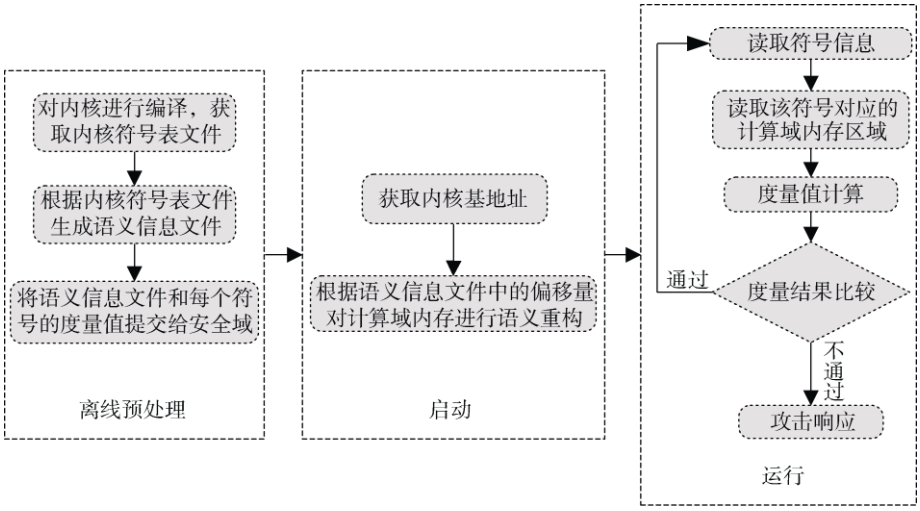

图 2 安全域内进行细粒度主动可信监测度量的流程

Figure 2 Process of fine-grained active trusted monitoring measurement in security domain

4.1.1 离线预处理阶段

离线预处理阶段为安全域提供了语义信息文件和初始度量值。在该阶段中，目标操作系统(该系统运行于计算域中)首先在离线状态下完成编译工作，获得目标操作系统的内核镜像文件以及内核符号表文件，从符号表文件中获得内核.text 段和.rodata 段的符号名、每个符号起始位置对应的计算域虚拟地址、符号所占内存空间的大小。然后将每个符号的计算域虚拟地址转换成计算域物理地址，并减去内核基地址(kernel_base)，即计算域内核加载到内存中的首地址，得到每个符号起始位置相对内核基地址的偏移量。每个符号的偏移量和其所占内存空间的大小可以确定每个符号对应的区域在计算域中的起始位置和结束位置。接下来对目标操作系统的每个符号分别进行度量以获取初始度量值，作为之后细粒度度量的度量基准值。符号名、相对偏移量、符

号所占内存空间的大小和初始度量值构成了计算域操作系统的语义信息文件。该语义信息文件将被提交给安全域，安全域在启动阶段以初始度量值为基准来验证目标操作系统的完整性。

4.1.2 启动阶段

启动阶段完成了对计算域内存进行语义重构的工作。安全域在启动过程中定位内核加载的基地址。在定位完成后，根据内核符号起始地址相对于内核基地址的偏移量和符号所占内存空间的大小，可以计算出每个符号起始位置和结束位置的计算域物理地址，进而实现内核关键数据结构的语义恢复。

4.1.3 运行阶段

计算域启动后，系统开始进入运行阶段，进行常规的计算任务。该阶段的主要目标是定期对计算域内核关键数据结构(如系统调用表，内核模块链表信息等)进行动态度量，重新计算该内核数据结构的度量值，将新的度量值与基准值进行校验，以发现针对内核的恶意攻击。

4.2 整体架构

图 3 是本文的基于安全优先架构的细粒度主动可信监测度量系统的架构图。细粒度度量工作主要由安全域中的五个部分完成，它们分别是：安全存储区、语义重构模块、主动监测模块、度量模块、攻击响应模块。从图 3 中可以看到，细粒度度量的过程中所有安全相关的任务都由安全域内的模块完成。这种架构通过硬件的隔离可以保证在安全域中运行的细粒度度量机制自身不受计算域攻击的影响。此外，安全域和计算域的 CPU 并行运行，将安全相关的任务完全交给安全域就不会占用计算域的资源，可以减少对计算域中应用进程的影响。

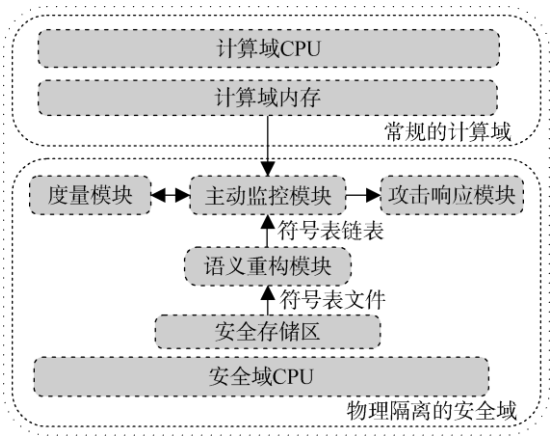

图 3 基于安全优先架构的细粒度主动可信监测度量系统

Figure 3 Fine-grained active trusted measurement monitoring system based on security priority architecture

4.3 架构分析

4.3.1 安全存储区

安全存储区与计算域物理隔离，是安全域可以访问，但计算域无法进行访问的存储区域。它对安全域的关键数据进行保护，防止计算域对其进行读写。细粒度度量过程中，安全存储区存储了计算域的语义信息文件(其中包括符号名及其对应的地址偏移量、所占内存空间的大小和度量基准值)，保证了这些信息不被计算域的攻击者篡改。

4.3.2 语义重构模块

语义重构模块负责完成对计算域内存中二进制数据语义信息重构的工作，包括安全域对计算域语义信息的解析和地址转换。语义信息的解析是指使用计算域提交到安全域的语义信息文件，从中解析出每个符号的名称、相对于内核基地址的偏移量、所占内存空间的大小、度量基准值等信息。地址转换是指将获取的计算域符号地址转换成安全域进行访问时可以使用的地址，进而在安全域实现对不同符号代表的计算域内存区域的访问和细粒度度量。

安全域的语义重构模块需要同时满足两个条件才能实现对计算域内存语义信息的解析：其一是安全域可以获得计算域的内核基地址(kernel_base)；其二是安全域可以获得内核符号起始地址和结束地址相对于内核基地址的偏移(offset)。

条件一在计算域操作系统使用内核地址随机化(kernel address space layout randomization , KASLR)时较难实现。使用 KASLR 来增强内核安全性时，计算机每次重新启动时内核基地址都会发生改变，安全域很难定位到计算域内核基地址。针对这一问题，本文提出两种方法，在保证计算域内核安全性不受削减(即保留 KASLR 机制)的前提下，同时保证安全域能够可靠、准确地定位计算域内核基地址。第一种方法对计算域内核进行轻量级修改(patch)操作，让计算域使用安全域提供的随机数作为内核基地址，这样安全域能准确定位到内核的基地址，同时保证计算域内核随机化机制不被破坏。但这种方法需要对计算域系统的内核代码进行修改。第二种方法是在预加载阶段对内核代码提取特征码并传入安全域，系统启动时安全域根据内核代码的特征码访问计算域内存进行匹配，匹配成功则找到内核基地址。这种方法不需要对内核代码进行修改。特征码匹配时需要进行内存查询的范围较小，采用恰当的方式选取特征码(如选取内核中位于每个页起始位置的一小段数据)可以减小特征码匹配的开销，在短时间内定位

到内核基地址。

针对条件二，本文在离线预处理阶段生成计算域操作系统的语义信息文件提交给安全域。该文件中存储了待度量的内核符号的起始地址相对于内核基地址的偏移量以及每个符号所占内存空间的大小。为了确定某一符号对应的待度量区域的位置，安全域需要得到该符号起始位置和结束位置对应的计算域物理地址。安全域在完成对计算域内核基地址的定位后，将其与语义信息文件中存储的该符号的起始地址相对内核基地址的偏移量相加，得到该符号在计算域中的起始物理地址，再将该物理地址与对应符号所占的内存空间大小相加，得到该符号结束物理地址。安全域按上述方式确定所有符号对应的待度量区域，进而实现内核关键数据结构的语义恢复。

为了完成安全域的语义重构工作，我们还需要对获得的符号地址信息进行一定的转换。安全域访问计算域指定位置读取要进行度量的数据时，不能直接使用计算域的虚拟地址或物理地址。计算域的虚拟地址不能在安全域中使用，因为使用计算域虚拟地址访问计算域内存时需要得到计算域的页表进行地址转换，安全域中没有计算域的页表信息。计算域的物理地址不能直接在安全域中使用，因为安全域有自己的内存，安全域中与计算域数值相同的物理地址并不代表同一个内存位置。

图 5 展示了从计算域虚拟地址到安全域虚拟地址的逐级转换过程。在对每个符号进行度量前，需要将其对应的计算域物理地址进行进一步转换，得到安全域中指向计算域内存正确位置的地址。这一转换是根据硬件提供的安全域对计算域内存的访问方式进行的。在整个细粒度可信度量的过程中，每个跨域的内存访问都需要进行这种地址转换。

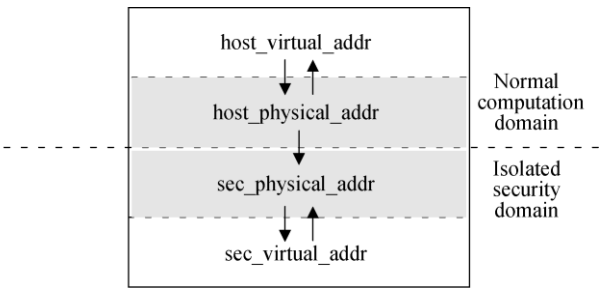

图 4　计算域虚拟地址到安全域虚拟地址的转换
Figure 4　Convers　ion from host_virtual_addr to sec_virtual_addr

4.3.3　主动监测模块

主动监测模块在系统运行过程中控制细粒度主动可信度量的进行。它顺序读取语义重构模块提供的符号信息列表，得到某符号的地址、大小和基准值信息后，访问对应的计算域内存区域，并获取该区域的数据。随后它将这些数据与该符号的度量基准值一起提供给度量模块进行度量值的计算和比较。如果返回结果显示度量值的比较通过，主动监测模块再读取下一个符号的信息；如果返回结果显示度量值的比较未通过，则主动监控控系统会调用攻击响应模块进行处理。

4.3.4　度量模块

度量模块实现具体的可信度量计算过程。它接收主动监测模块提供的某符号的二进制数据和度量基准值，对二进制数据进行度量值的计算，并将度量值与该符号的度量基准值相比较，返回比较结果。度量值的计算在安全域中进行，计算域无法对其造成影响。

4.3.5　攻击响应模块

攻击响应模块负责监测攻击行为之后的处理工作。如果度量值与度量基准值的比较结果不一致，则说明被度量的位置被更改，主动监测模块会将这次度量的符号的相关信息传递给攻击响应模块进行处理。

得到监测结果后，可以据此对计算域受到的攻击进行分析。不同类型的攻击行为得到的监测结果有不同特征。表 1 总结了 Rootkit 中常见的攻击行为、具有该行为的 Rootkit 实例以及可以用来实现该攻击行为的符号名，如果细粒度监测度量发现某符号名对应的位置被篡改，则可以根据表 1 中总结的被篡改的符号名与攻击行为的对应关系进行分析。如果检测到的被篡改的符号可实现的攻击行为未知，可以结合内核代码等信息进行进一步推测。

攻击响应模块还可以通过安全域硬件提供的功能来控制计算域，如断开网络、禁用外设，甚至停止计算域系统的运行。它还可以通过硬件将检测到的攻击信息直接从安全域传送到外部，不经过计算域传输，这样可以防止攻击监测结果被计算域的攻击者篡改。

5　系统实现

5.1　离线预加载阶段的语义信息收集

在离线预加载阶段，我们需要在计算域内进行内存语义信息的收集。在 2.6 内核之后，为了更好地调试内核，Linux 引入了符号表。内核中的符号表提取内核使用的所有全局函数、静态函数和非堆栈数据变量的地址，并以数组形式存储在内存中，作为

内核中的只读数据。相当于在内核中存储一个system.map。

表 1 Rootkit 攻击行为与实现方式

Table 1 Rootkit's Attack Behavior and Implementation

攻击行为	Rootkit 名称	被篡改的符号名
隐藏文件/目录	AFkit	SyS_open proc_root_readdir
	Suterusu	SyS_getdents
	Reptile	SyS_getdents64
	Puszek	ext4_readdir
	enyelkm	filldir
	Diamorphine	filldir64
	CSE509-Rootkit	sys_stat sys_lstat
隐藏进程	Suterusu	proc_root_readdir SyS_getdents
	Reptile	SyS_getdents64
	CSE509-Rootkit	find_task_by_vpid next_tgid
隐藏连接	Suterusu	tcp4_seq_show udp4_seq_show
	Reptile	udp6_seq_show
	Puszek	tcp6_seq_show
隐藏网卡模式	Suterusu	SyS_open dev_get_flags
密码记录	Puszek	SyS_sendto
隐藏/篡改文件内容	Reptile enyelkm	vfs_read
获取 root 权限	enyelkm	SyS_kill
	CSE509-Rootkit	sys_setuid

内核同时提供了一系列导出函数来实现对符号表数据结构的查找和遍历操作。本实验中使用内核中导出的函数进行内核符号表数据结构的遍历，获取每个符号的起始地址和所占空间大小，然后与内核加载的起始地址相减，得到每个符号相对内核起始地址的偏移。鉴于 sys_call_table 中存储的函数指针在实际攻击中经常被修改，实验中将这一符号的内容再次进行拆分。sys_call_table 的内容相当于指向一系列系统调用函数的指针数组。在实验中，64 位操作系统的指针占 8bytes，所以我们将每 8bytes 数据作为一个指针，进一步获取每个指针指向的系统调用函数名称。

在计算域中收集到的语义信息需要按照约定的格式来存储，安全域只有接收以约定格式存储的语义信息文件才能进行正确的解析。安全域需要得到内核的符号名称、相对内核起始地址的偏移、符号所占空间的大小以及每个符号在离线阶段的度量值。上述信息在离线预处理阶段收集并存储在一个文件中，每一行代表一个符号的相关信息，每一条符号信息包含的 4 项内容按固定顺序存放。这样在语义信息文件传入安全域后，安全域按照同样的格式进行解析就可以获得所需要的语义信息。

5.2 启动阶段的语义信息重构

在离线预加载阶段，安全域已经获得每个符号相对内核加载起始地址的偏移，还需在启动时得到内核加载的起始物理地址，才能得到每个符号在计算域内存中的正确位置。在由安全域指定随机化的内核基地址的方法中，由于实验需在支持安全优先架构的特殊计算机中进行，更改操作系统代码的程序较为繁琐。在特征码匹配内核基地址的方法中，需要额外的研究和实验来衡量选取的特征码是否在保证准确率的基础上有较低的匹配耗时。由于获取内核基地址的过程并不影响后续的主动度量，考虑到上述问题，实验中对启动阶段内核基地址的获取进行了一定的简化。在实验中，我们利用内核配置选项，令内核代码每次都加载到同一物理地址。在对语义信息文件进行解析时，将每个符号对应的偏移量加上一个固定的地址，便可得到符号在计算域中的正确物理地址。

安全域获得计算域的语义信息文件和内核加载起始地址后，便有足够的信息对计算域内存二进制数据进行语义重构。本实验中，安全域读取语义信息文件，按照预先约定的格式对其进行解析和地址转换，将其中的数据存储为符号信息链表(如图 6 所示)。在解析语义信息文件的过程中，根据实验环境，安全域将每个符号的偏移量加上固定的计算域内核加载起始地址，得到每个符号在计算域内的起始物理地址。然后根据安全域和计算域的物理地址映射关系，在安全域中将每个符号的计算域物理地址加上一个由硬件决定的地址偏移，将其转换成安全域对其访问时应使用的地址进行存储。该链表每个节点中存储的具体内容如下:

（1）Nam: 计算域中每个符号的符号名称;

（2）Addr: 每个符号的计算域起始物理地址对应的安全域地址;

（3）Size: 每个符号所占的空间大小;

（4）Benchmark: 每个符号对应的可信测量基准值;

（5）Prev、next: 指向前驱结点和后继节点的指针。

链表可以在不确定符号表文件存储的符号总数时以较少的空间消耗存储数据，为了实现上的方便我们采用该方法进行存储。

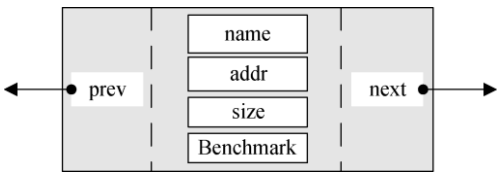

图 5　存储符号信息的数据结构

Figure 5　Data structure for storing symbol information

在符号信息链表构建完成后，安全域对语义信息文件的解析和符号地址的转换工作完成，安全域可以分别对每个符号所在的计算域内存区域进行访问，获取对应的二进制数据并进行细粒度度量。

5.3　运行时细粒度可信度量监测的实现

为了通过度量结果正确监测到攻击者对内存的篡改，本系统的度量模块使用 MD5 消息摘要算法[21]对每个符号对应的二进制数据信息生成信息摘要作为度量值。不同的二进制数据得到的 MD5 散列值不同。这一特性可以用于确保信息的完整性和一致性。MD5 算法计算每个符号对应的度量值的过程在安全域中进行，计算域无法影响其计算结果。

每个符号都需要一个度量基准值与单次度量值做比较，来确保本次被度量的二进制数据与安全状态下的二进制数据相比未发生改变。本实验中将度量基准值设置为在离线预加载阶段使用 MD5 消息摘要算法得到的每个函数的度量值。该值将会与符号的其他信息一起存入符号信息链表，并在安全存储区中存储其备份。

6　实验

本文的实验在使用 64 位中标麒麟操作系统的台式机上进行。该机器的内存为 4Gb，使用兆芯处理器，并且支持 SOC(System-on-chip)来实现安全优先架构。我们在其上实现了细粒度监测度量系统，并对其进行了攻击测试和性能测试。

6.1　攻击测试

本节我们在安装了细粒度度量的机器上运行了一些 Rootkit，以测试细粒度可信监测度量系统的效果。这些 Rootkit 对内核的.rodata 和.text 段的不同内容进行了攻击。由于不同 Rootkit 的实验结果有重复，我们结合表 1 列出的 Rootkit 的攻击方式和攻击行为，在表 2 中列出其中具有代表性的三个 Rootkit 的实验结果。

AFKIT 是 Github 上的一个开源反取证 Rootkit，它使用系统调用劫持的技术。Suterusu 是一个经典的 Linux Rootkit，它通过修改目标函数的序言将执行流转移到替换例程来进行 hook，而不是 hook 系统调用表。此外，我们还开发了一个名为“rodata_Rootkit”的 Rootkit 来模拟以前没有发现的新攻击。这一 Rootkit 的功能是篡改记录 USB 设备访问信息的字符串，妨碍系统对 USB 设备的审计工作。

表 2　攻击测试的实验结果

Table 2　Experimental results of attack tests

Rootkit 名称	粗粒度度量	细粒度度量
AFkit	.rodata	SyS_open SyS_kill SyS_getdents SyS_chdir SyS_getdents64 proc_root_readdir
Suterusu	.text	ext4_readdir dev_get_flags tcp4_seq_show udp4_seq_show inet_ioctl udp6_seq_show tcp6_seq_show
rodata_rootkit	.rodata	dcbnl_rtnl_policy

(注: 第二列和第三列是使用两种度量方式得到的被修改内容所在的地址范围。其中 SyS 开头的是系统调用表中的指针指向的系统调用)

攻击测试的实验结果见表 1。从这个表中可以看出，细粒度测量监测系统能够以符号级的精度定位被攻击的内核代码段和只读数据段的位置。我们可以利用这些信息进一步分析攻击者的行为和目的，并且对系统的恢复工作提供信息。

从监测结果来看，AFkit 主要使用了修改 sys_call_table 的方法进行攻击，我们可以再进一步查看 sys_call_table 中何处被修改。SyS_open 用来打开一个文件，hook 该函数可能是为了阻止用户打开系统关键文件或目录，如设备文件/dev/port 等。SyS_kill 在 Linux 系统中用来终止一个进程，在 AFkit 中 hook 该函数是为了根据传入的信息进行攻击行为的选择。SyS_getdents 和 SyS_getdents64 用来查询文件或者目录，hook 这两个函数一般是为了在用户使用 ls、ps 等命令时隐藏文件。SyS_chdir 用来更改当前目录，实现了 cd 命令的功能，hook 它是为了防止用户使用 cd 命令进入某类目录项。

Suterusu 为了避免被杀毒软件检测到，并未使用修改 sys_call_table 的方法。细粒度度量的实验结果显示，它修改了 8 个.text 段的函数在内存中的内容，这几个函数分别对应了 Suterusu 的几个功能：proc_root_readdir 用来读取/proc 目录下文件，在使用

ps、top 之类的命令查看当前运行进程时, 会调用该函数对/proc 目录进行遍历得到所有当前运行进程的 pid, 它被篡改意味着 Suterusu 可能提供了隐藏某 pid 的功能; ext4_readdir 负责读取 ext4 类型文件系统的目录, 这一篡改可能是为了隐藏文件; dev_get_flags 在系统中用来获取网络接口标识, Suterusu 改动该函数是为了实现隐藏网卡的混杂模式的 PROMISC 标识; tcp4_seq_show、udp4_seq_show、udp6_seq_show 、tcp6_seq_show 这四个函数分别用来在/proc 文件系统中生成 TCP4、UDP4、UDP6、TCP6 连接的相关内容, 篡改他们一般是为了隐藏对应类型的连接信息; inet_ioctl 是一个很少被使用的控制函数, 如果没有监测结果的提醒, 防御者很难在没有源码的情况下发现攻击者对它进行了修改。在 Suterusu 中, 作者篡改该函数是用来与用户进程通信, 在恶意进程中通过运行时传入的参数来进行功能的选择。

无论是粗粒度度量还是细粒度度量, 都成功监测到了我们开发的 rodata_rootkit, 这证明了主动可信防御体系在未知攻击监测上的有效性。细粒度度量的实验结果显示, rodata_rootkit 对.rodata 段的 dcbnl_rtnl_policy 进行了篡改。该处存储的是一个可读的字符串, 只要将其中存储的内容与正常系统对比, 就可以发现攻击者对它做了什么修改。而粗粒度度量只能确定修改发生在.rodata 段, 对于防御者来说范围太大, 很难根据该检测结果对攻击者的意图进行分析。这一结果表明在对未知攻击方式和攻击目的的 Rootkit 进行分析时, 使用细粒度监测度量有更好的效果。

6.2 性能测试

本节我们使用一个名为 lmbench[22]的工具来测试计算域主机的性能。Linux 性能测试工具 Lmbench 是一套简易可移植的、符合 ANSI/C 标准的、为 UNIX/POSIX 而制定的微型测评工具。它的测量内容可以分为两个方面: 响应时间和带宽。我们分别在没有装任何可信监测系统的计算机和装有本文提出的细粒度可信监测度量系统的计算机上运行了 lmbench 并记录了结果。测试结果包括以下几个方面:

6.2.1 与进程相关的操作

该项测试包括执行 getppid 所用时间、从/dev/zero 读一个字节的时间和写一个字节到/dev/null 的时间的平均值、得到一个文件信息以及打开一个文件再关闭它所需的时间、通过 TCP 网络连接选择 100 个文件描述符所耗用的时间、信号的处理程序安装和捕捉所需时间、进程的创建和执行所需时间。这一部分的测试结果见表 3。

从表 3 的数据中我们可以看出, 与进程相关的大部分操作都没有受到明显影响。前七项的结果较小, 其差值范围为 0~0.03ms, 最大差值约占无可信监测情况下测量结果的 0.6565%。后三项测量值较大, 测量差值也较大, 范围为 1.6015~104.5ms, 其中 sh proc 项的差值最大, 约占无可信监测情况下测量结果的 2.7511%。后三项涉及到的内存读写操作较多, 在性能上受到的影响更明显。据此可以推测细粒度度量对于内存读写操作的影响比对其他操作的影响更大。

表 3 与进程相关的操作所用时间(ms)

Table 3 Time spent on process-related operations

	无主动可信监测	使用细粒度可信监测	差值
stat	2.26	2.27	0.01
open close	4.57	4.60	0.03
slct TCP	4.81	4.82	0.01
sig inst	0.37	0.37	0
sig hndl	1.77	1.78	0.01
fork proc	286.8722	288.4737	1.6015
exec proc	1083.2000	1086.5000	3.3000
sh proc	3798.5000	3903.0000	104.5000

(注: 差值是每一行的测试项在使用细粒度可信监测的环境下得到的结果减无主动可信监测的环境中得到的结果)

6.2.2 上下文切换(切换进程加上恢复进程所有状态所用的时间)

该项测试中, 2p/0K 项代表每个进程的 size 为 0(不执行任何任务), 进程数为 2 时上下文切换耗用的时间, 2p/16k 项代表每个进程的 size 为 16k(执行任务), 进程数为 2 时上下文切换时耗用的时间, 以此类推。测试结果如表 4 所示。

两种环境下耗时的差值范围为–0.2~0.6ms, 分别为无主动可信监测环境的–1.8182%和 3.8217%。可以看到在进程较少、进程大小较小的时候, 细粒度可信监测对进程上下文切换的影响不大。在进程数相同时, 进程越大, 细粒度可信监测对进程上下文切换的影响表现得越明显。通过上述分析我们可以看到, 影响上下文切换时间的因素主要是对内存的访问, 需要的内存存取操作越多, 上下文切换受到的影响越明显。

表 4 上下文切换所用时间(ms)

Table 4 Time spent on context switching

	无主动可信监测	使用细粒度可信监测	差值
2p/0K ctxsw	5.2900	5.2800	–0.01
2p/16K ctxsw	11.0	10.8	–0.2
2p/64K ctxsw	12.6	12.8	0.2
8p/16K ctxsw	11.8	11.7	–0.1
8p/64K ctxsw	14.0	14.2	0.2
16p/16K ctxsw	12.1	12.0	–0.1
16p/64K ctxsw	15.7	16.3	0.6

(注: 差值是每一行的测试项在使用细粒度可信监测的环境下得到的结果减无主动可信监测的环境中得到的结果)

6.2.3 基本整数运算、单精度浮点数运算、双精度浮点数运算

基本整数运算测试包括位运算和加、乘、除、求模运算, 该项测试结果见表 5。基本单精度浮点数运算测试包括加、乘、除运算和 bogomflops, 该项测试结果见表 6。基本双精度浮点数运算测试包括加、乘、除运算和 bogomflops, 该项测试结果见表 7。

从表 5~表 7 中可以看到, 两种不同环境下的测量结果没有太大变化。与计算相关的操作主要受 cpu 性能的影响, 本节的实验结果表明细粒度可信监测度量系统对 cpu 运算性能影响很小。

表 5 基本整数运算所用时间(ns)

Table 5 Time spent on basic integer operations

	无主动可信监测	使用细粒度可信监测
intgr bit	0.5000	0.5000
intgr add	0.0600	0.0600
intgr mul	1.0500	1.0500
intgr div	11.8	11.8
intgr mod	10.5	10.5

表 6 基本单精度浮点数运算所用时间(ns)

Table 6 Time spent on float operations

	无主动可信监测	使用细粒度可信监测
float add	1.5100	1.5100
float mul	1.5100	1.5100
float div	6.7800	6.7800
float bogo	6.5700	6.5700

表 7 基本双精度浮点数运算所用时间(ns)

Table 7 Time spent on double operations

	无主动可信监测	使用细粒度可信监测
double add	1.5100	1.5100
double mul	2.0200	2.0100
double div	10.3	10.3
double bogo	10.1	10.1

6.2.4 文件&VM 系统延时

该项测试包括 0k 和 10k 文件创建/删除所花费的时间、mmap latency(将指定文件的开头 n 个字节 map 到内存, 然后执行 umap, 并记录每次 map 和 umap 共耗用的时间)、保护页和缺页延时的时间、100fd select(对 100 个文档描述符配置 select 的时间)。测试结果见表 8。

从表 8 的数据中我们可以看到, 文件大小更改后文件创建和删除的测量差值没有变化, 但两个环境下的耗时都有增加, 文件创建的测量结果差值占无主动可信监测环境下的测量结果比例分别为 0.9070%和 0.5384%。mmap latency 的差值为 66ms, 是无主动可信监测情况下的 1.068%。prot fault 项的测试结果在两种环境下的差值为 0.007, 占无主动可信监测情况下的 1.5086%。page fault、100fd selct 两项测量值的差值较小, 可以忽略。从表 8 的数据分析, 对于经常进行文件的增删和拷贝的程序, 细粒度度量会对其性能有一些影响, 但影响程度不大。

表 8 文件&VM 系统延时(ms)

Table 8 File & VM system latencies

		无主动可信监测	细粒度可信监测	差值
0k File	create	44.1	44.5	0.4
	delete	17.2	17.2	0
10k File	create	74.3	74.7	0.4
	delete	25.5	25.5	0
mmap latency		6180.0	6246.0	66
prot fault		0.464	0.471	0.007
page fault		1.52550	1.52170	–0.0038
100fd select		2.332	2.331	–0.001

(注: 差值是每一行的测试项在使用细粒度可信监测的环境下得到的结果减无主动可信监测的环境中得到的结果)

6.2.5 本地通信带宽

该项测试包括进程间使用各种方式进行通信的带宽、文件和内存的重复读取速率、内存拷贝以及内存读写到 processor 的带宽。测试结果见表 9。

这一系列测试中受影响最明显的是 bcopy(hand)这项, 约差 6.773%。鉴于该次实验数据是整个性能测试中唯一差值百分比超过 5%的测试项, 我们收集了未安装度量程序的环境下表 9 的三次实验结果, 其中 bcopy(hand)结果见表 10。可以看到这一项的测试结果有浮动, 在三次实验中与使用细粒度可信监测时测量值的差值百分比绝对值的最小值达到了 4.4403%, 三次实验自身最大值和最小值的差值百分比为2.4415%, 所以虽然细粒度可信监测造成了性能

的损耗，但考虑到该测试项明显受到机器中其他因素的影响，实际由细粒度监测度量引进的性能损耗与系统正常情况下的波动造成的运行差异差不多，最终的差值是两者影响的叠加。

表 9　本地通信带宽(Mb/s)

Table 9　File & VM system latencies

	无主动可信监测	使用细粒度可信监测	差值	差值百分比
pipe	1648	1643	–5	–0.3034%
file reread	2222.1	2206.6	–15.5	–0.6975%
mmap reread	5316.6	5311.5	–5.1	–0.0959%
bcopy(libc)	1795.3	1779.8	–15.5	–0.8634%
bcopy(hand)	1687.5	1573.2	–114.3	–6.7733%
mem read	4006	3982	–24	–0.5991%
mem write	2071.36	2051.65	–19.71	–0.9515%

(注: 差值=使用细粒度可信监测的环境下的测试结果–无主动可信监测的环境下的测试结果，差值百分比=差值/无主动可信监测的环境下的测试结果)

表 10　bcopy(hand)测试结果(Mb/s)

Table 10　Test results of bcopy (hand)

	无主动可信监测	差值	差值百分比
1	1687.5	–114.3	–6.7733%
2	1662.0	–88.8	–5.3430%
3	1646.3	–73.1	–4.4403%

(注: 此处差值=表 9 中使用细粒度可信监测的测试结果–每行的无主动可信监测环境下的测试结果，差值百分比=差值/无主动可信监测环境下的测量结果)

6.2.6　性能测试总结

图 6 展示的是每种类型的性能测试在两种测试环境下测试结果差值百分比的平均值的绝对值。在前文我们根据测试内容将性能测试分为五类，每一类中有若干测试项。对于每一个测试项，我们都使用细粒度可信监测的环境下的测试结果减无主动可信监测的环境下的测试结果，得到二者的差值，然后将差值除以无主动可信监测的环境下的测试结果，得到每个测试项的差值百分比。对于每一类测试，我们将其中的测试项的差值百分比相加，除以测试项个数，然后取绝对值，得到每种类型的性能测试在两种测试环境下测试结果差值百分比的平均值的绝对值。

通过这张图我们可以看出，两种测试环境下不同类型的测试受到的影响程度不同。对于 cpu 密集型的操作，如基本运算的用时，两种测试环境几乎没有差别；而对于内存访问密集型的测试，如本地通信带宽，则影响较明显。我们分析这是由于在内存访问中，如果计算域的主机端恰好与安全域读写同一块内存区域，那么后访问的一方需要等待先访问的一方操作完成。但总体上看，细粒度监测度量系统对主机运行性能的影响较小，对各种类型的应用软件都能提供较好的支撑。

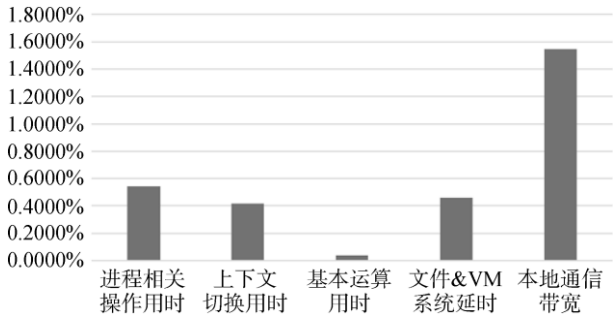

图 6　两种测试环境下测试结果差值百分比的平均值的绝对值

Figure 6　Absolute value of the average of the percentage of difference of the results between the test in two test environments

7　结论

针对当前主动可信度量存在的度量结果粒度较粗的问题，本文提出了一种基于安全优先架构的细粒度主动可信监测度量方法，提升了主动度量的精度。通过符号信息的可靠传递，本文克服了硬件隔离导致的安全域获取计算域语义信息的困难，在安全域中实现了对计算域内存的符号级别的细粒度度量，提升了基于安全优先架构的主动可信监测度量结果的精度。实验表明，本文提出的细粒度主动可信监测度量方法可以成功实现对计算域内存进行符号级别的监测，同时对计算域性能的影响较小。未来，我们可以将细粒度度量的范围扩展到内核模块，以进一步提高整个系统的安全性。

参考文献

[1] Donghyun Kwon, Kuenwhee Oh, Junmo Park, Seungyong Yang, Yeongpil Cho, Brent ByungHoon Kang, and Yunheung Paek, “Hypernel: A Hardware-Assisted Framework for Kernel Protection Without Nested Paging” *Proceedings of the 55th Annual Design Automation Conference (DAC)*. ACM, pp 34:1-34:6, 2018.

[2] Yang Penglin, Tao Limin, and Wang Haitao: “Rttv: a dynamic CFI measurement tool based on tpm”. *Information Security(IET)*, vol. 12, no. 5, pp. 438-444, 2018.

[3] Wang Yong, Zhang Yuhan, Hong Zhi, Wen Ru, Fan Chengyang, and Wang Juan: “Kernel integrity measurement architecture based

on tpm 2.0". *Computer Engineering*, vol. 44, no. 3, pp. 166-170, 2018.
(王勇,张雨菡,洪智,文茹, 樊成阳, 王娟, “基于 TPM2.0 的内核完整性度量框架”, *计算机工程*, 2018, 44(3):166-170。)

[4] Dan, Meng, Rui Hou, Gang Shi, Bibo Tu, Aimin Yu, Ziyuan Zhu, Xiaoqi Jia, and Peng Liu, “Security-first architecture: deploying physically isolated active security processors for safeguarding the future of computing.” *Cybersecurity*, vol. 1, no. 1, pp. 2, 2018.

[5] “TPM 1.2 Main Specification” Trusted Computing Group, https://trustedcomputinggroup.org/resource/tpm-main-specification/, Mar. 1, 2011.

[6] Arthur Will, Challener David, and Goldman Kenneth, “A Practical Guide to TPM 2.0” 2015.

[7] Liqun Chen, and Rainer Urian, “Algorithm agility – discussion on tpm 2.0 ecc functionalities” , 2016.

[8] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian, “One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation” *In 2017 IEEE Symposium on Security and Privacy*, pp. 901-920, 2017.

[9] Zhang Xing, and Shen Changxiang, “A Novel Design of Trusted Platform Control Module” *Geomatics & Information Science of Wuhan University*, vol. 33, no. 10, pp. 1011-1014, 2008.
(张兴, 沈昌祥, “一种新的可信平台控制模块设计方案”, *武汉大学学报·信息科学版*, 2008, 33(10): 1011-1014。)

[10] Huang Jianhui, Shen Changxiang, and Xie Wenlu, “The tpcm 3p3c defense architecture of safety and trusted platform” *Journal of Wuhan University*, vol. 64, no. 2, pp. 109–114, 2018.
(黄坚会, 沈昌祥, 谢文录, “TPCM 三阶三路安全可信平台防护架构”, *武汉大学学报·理学版*, 2018, 64(2): 109-114。)

[11] 余祥, 张健, 李强, “基于 TPCM 的终端数据泄露防护技术研究”,*第六届中国指挥控制大会论文集(下册)*, 2018。

[12] Xiaoqi Jia, Yun He, Xiyao Wu, and Huiqi Sun, “Performing trusted computing actively using isolated security processor” *In Proceedings of the 1st Workshop on Security-Oriented Designs of Computer Architectures and Processors*, pp. 2–7, 2018.

[13] Nick L. Petroni Jr. , Timothy Fraser, Jesus Molina, and William A. , “Copilot - a coprocessor-based kernel runtime integrity monitor” *Proceedings of Usenix Security Symposium*, pp. 179–194, 2004.

[14] Ziyi Liu, JongHyuk Lee, Junyuan Zeng, Yuanfeng Wen, Zhiqiang Lin, and Weidong Shi, “Cpu transparent protection of os kernel and hypervisor integrity with programmable dram” *in The 40th Annual International Symposium on Computer Architecture*. ACM, pp. 392-403, 2013.

[15] Hyungon Moon, Hojoon Lee, Jihoon Lee, Kihwan Kim, Yunheung Paek, and Brent Byunghoon Kang, “Vigilare: toward snoopbased kernel integrity monitor” *in Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, pp. 28–37, 2012.

[16] Ahmed M. Azab, Ning Peng, Wang Zhi, Xuxian Jiang, Xiaolan Zhang, and Nathan C. Skalsky, “Hypersentry: Enabling stealthy in-context measurement of hypervisor integrity” *in Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 38-49, 2010.

[17] Brian Delgado, and Karen L. Karavanic, “EPA-RIMM: A framework for dynamic SMM-based runtime integrity measurement” *CoRR*, vol. abs/1805.03755, 2018.

[18] Wang Jiang, Angelos Stavrou, and Anup Ghosh, “Hypercheck: A hardware-assisted integrity monitor” *in Recent Advances in Intrusion Detection, 13th International Symposium*, pp. 158-177, 2010.

[19] Fengwei Zhang, Kevin Leach, Kun Sun, and Angelos Stavrou, “SPECTRE: A dependable introspection framework via system management mode” *in 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems & Networks(DSN)*, pp. 1-12, 2013.

[20] “System management mode” https://en.wikipedia.org/wiki/ System_Management_Mode, Jul.10, 2019.

[21] Ronald L. Rivest, “The MD5 message-digest algorithm,” *RFC*, vol. 1321, pp. 1-21, 1992.

[22] McVoy Larry and Staelin Carl, “Imbench: Portable tools for performance analysis,” *in Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, pp. 23, 1996.

田 竞 于 2017 年在南开大学计算机科学与技术专业获得学士学位。现在中国科学院大学网络空间安全专业攻读硕士学位。研究领域为信息安全。研究兴趣包括: 可信计算、操作系统安全等。Email: tianjing1996@iie.ac.cn

孙慧琪 于 2017 年在哈尔滨工业大学软件工程专业获得工学硕士学位。现任中国科学院信息工程研究所助理研究员。研究领域为信息安全。研究兴趣包括：可信计算、操作系统安全等。Email: sunhuiqi@iie.ac.cn

武希耀 于 2015 年在中南大学软件工程专业获得硕士学位。现任中国科学院信息工程研究所助理研究员。研究领域为系统安全。研究兴趣包括：内存完整性检测、RootKit 检测等。Email: wuxiyao@iie.ac.cn

贾晓启 于 2010 年在中国科学院研究生院信息安全专业获得博士学位。现任中国科学院信息工程研究所研究员。研究领域为系统安全。Email: jiaxiaoqi@iie.ac.cn

张伟娟 于 2018 年在中国科学院信息工程研究所信息安全专业获得博士学位，现任信息工程研究所助理研究员。研究领域为云计算安全。Email: zhangweijuan@iie.ac.cn

黄庆佳 博士，高级工程师。主要从事操作系统安全、云计算安全、高级威胁检测、恶意代码分析等研究工作。Email: huangqingjia@iie.ac.cn