

文章编号:1001-9081(2006)12-2916-03

基于度量模块的入侵检测模型的研究与实现

林涛¹, 胡华平², 张怡², 刘波²

(1. 公安部第八局 信息处, 北京 100031; 2. 国防科学技术大学 计算机学院, 湖南 长沙 410073)
(lintao5288@163.com)

摘 要: 提出了一种借鉴可信计算概念的基于度量模块的底层入侵检测的方法, 借助访问控制等技术, 在现有 PC 体系结构下提出了针对硬件固件的入侵检测模型, 对硬件固件攻击和注入行为的检测进行了研究, 并给出了模型验证结果。

关键词: 硬件固件; BIOS; 入侵检测

中图分类号: TP309.1 **文献标识码:** A

Study and implementation of the intrusion detection model based on measurement module

LIN Tao¹, HU Hua-ping², ZHANG Yi², LIU Bo²

(1. 8th Bureau, Ministry of Public Security, Beijing 100031, China;

2. School of Computer Science, National University of Defense Technology, Changsha Hunan 410073, China)

Abstract: Based on Trusted Computing concept and RBAC technology, this paper proposed a firmware-level intrusion detection model which challenged current PC architecture named measurement module and implemented the key steps.

Key words: hardware firmware; BIOS; intrusion detection

0 引言

随着互联网的高速发展,对安全保障提出了更高的要求。人们需要系统能够保证足够真实性、完整性、隐私保护、匿名访问控制和扩展能力。尽管密码技术和一些其他的安全手段如防火墙、IDS、反病毒软件给出了一系列的解决方案,但是它们的正确运行都是建立在下层系统、尤其是操作系统的安全之上的,而操作系统的安全又建立在硬件固件的安全之上。各种平台的安全问题很多是由于其自身体系结构的薄弱和复杂性而造成的。针对这一问题,主要有两大研究组织:可信计算集团 Trusted Computing Group(TCG)提出的可信计算模块(TPM)和微软公司提出的下一代安全计算基(Next Generation Secure Computing Base, NGSCB)。前者提出的是一个硬件平台规范,可是实现跨平台^[1,2]而后者是特定的适用 Windows 操作系统的硬件和软件相结合的方案。TCG 和 NGSCB 的工作原理都是基于密码体系,通过信任的传递和确认来维护计算的安全根基。

可信计算可以提供对数字版权的保护,保证软件的完整性、合法性还可以防止软件被病毒或者恶意攻击篡改后加以利用,利用基于可信计算的身份认证、访问控制、信息封存和完整性校验等技术才能真正提供细粒度的软件保护模型。可信平台模块 TPM(Trusted Platform Module)作为可信计算平台的重要部件,通常要求和计算机硬件集成并和核心信任度量根 CRTM(Core Root of Trust for Measurement)一起组成信任基 TCB(Trusted Computing Base)。但是现有的硬件体系结构和操作系统均不支持可信计算,如果建立基于可信计算的可信 PC,意味着对现有软硬件体系结构进行较大改动,这也是可

信计算机推广遇到困难的主要原因。但是,可信计算的体系结构给我们提供了从系统底层确保系统安全的模型:确保系统加电之后系统的逻辑执行流的信任是可以度量的。如何保证硬件固件的可信性,现在还没有清晰的思路。

随着现代操作系统的发展,操作系统安全级别不断提升,对用户态(User Mode)程序权限的限制也不断加强,硬件固件,即具有软件功能的硬件,在现代的计算机系统之中,一般认为它是硬件存储其初始化、外围控制的程序代码的只读记忆体(ROM)。只读存储器包括:可擦写编程内存储器(EPROM)和快速只读存储器(Flash ROM)等,其中快速只读存储器是当今主板、显卡以及手机等便携设备存放监控程序的主要器件。随着技术的发展,现在更倾向认为固件是一种软件。只读存储器在一般状态下,先前写入的数据无法被擦除、破坏,即使关掉电源之后内容仍旧存在;在特殊应用程序的配合以及相关硬件设置之后,存放在快速只存储器中的数据、代码和设置信息能够被擦除和修改。针对不同的硬件,各种固件名称也各不相同。主板的固件即 BIOS(Basic Input and Output System)芯片,而针对光驱显卡等硬件固件有时也称为 Firmware。文献[3,4]介绍了快速只读存储器的更新和优化技术,这种功能和可用软件更新的特点,为基于固件的操作提供了可行性。当写入硬件固件的内容是“精心构造”的正确运行的代码时,称之为注入;相反,如果写入固件的内容是“精心构造”的错误代码时,这就形成了攻击。直接针对 Windows 2000/XP/2003 平台的硬件固件的攻击和瘫痪技术的检测公开发表的资料非常少,相关技术和文献主要以主板 BIOS 芯片为主,文献[5]介绍了 BIOS 芯片硬件设置和部分版本的源代码,文献[6,7]中提出了 AWARD 系列 BIOS 芯片的

收稿日期:2006-06-09;修订日期:2006-08-18

基金项目:国家自然科学基金资助项目(60573136);国防预研基金资助项目(51419020105KG0110)

作者简介:林涛(1982-)男,安徽人,硕士研究生,主要研究方向:网络与信息安全; 胡华平(1967-)男,福建人,博士生导师,主要研究方向:信息安全与密码学; 张怡(1973-)女,四川人,副教授,博士,主要研究方向:网络与信息安全; 刘波(1973-)男,湖北人,博士研究生,主要研究方向:网络与信息安全。

小型应用程序注入和逆向工程方法。这种针对硬件固件的注入方法从底层破坏了系统的可靠性,由于操作系统此时没有加载,针对这种注入的检测也无从谈起,因此需要一个在不改变现有 PC 体系结构下入侵检测模型从底层加固系统的安全。

1 基于度量模块的入侵检测模型

1.1 度量模块的组成

本文提出的度量模块(Measurement Module)入侵检测模型结构如图 1 所示。该模型是以软件方式实现的度量模块,借助可信计算的完整性验证、口令验证、基于角色的访问控制以及备份的关键技术,从多层次和多角度保护了系统加电之后逻辑执行流的正确性,防止被非法修改和使用。模型利用口令认证机制,通过对重要数据的备份和加密,防止攻击者和恶意代码破译和越权使用,为完整性恢复提供了基础。

用户	基于口令的认证		基于RBAC访问控制	
程序	完整性检查	备份	完整性恢复	报警
度量模块(Measurement Module)				

图 1 基于度量模块的入侵检测模型

度量模块使用状态数据库来保存各个模块的正确状态,完整性检查在系统运行之初,建立被保护系统的安全的初始状态档案。这里的初始状态是指被保护系统已经安装配置完毕,但尚未与网络连接或未启动网络服务时的状态。此时系统的数据来源比较单一,易于保证其安全性。初始状态就是系统正确的安全状态。系统一些正常的操作(安装删除程序,增减用户,其他必要的管理)可能会使系统的状态发生变化,这就要系统的安全状态同步更新。为此,考虑使用数据库保存系统的正确状态。采用数据库可以使记录的保存结构化、迅速索引和定位、方便检查和更新。数据结构的设计应包含尽可能多的文件属性,如名称、版本、大小、权限、散列值等。同时合理的数据结构也能保证在检查时更新相应的参数易于实现。当完整性检查发现数据不一致时,报警模块启动,在基于用户角色访问控制的条件下开始恢复系统遭到篡改的数据和资料。

1.2 度量模块的工作原理

图 2 给出了度量模块的工作过程,在进入工作状态后,通过口令验证和角色控制之后,度量模块开始对 BIOS 启动模块、BIOS 可选模块、操作系统加载器、操作系统引导程序和关键系统驱动分别进行读取操作,进行签名后和数据库中保存的资料进行对比,签名吻合则说明系统硬件固件和操作系统加载没有受到攻击和注入,签名不吻合则说明硬件固件遭到非法篡改,度量模块开始定位遭到篡改部位并且报警,恢复正确数据和资料。

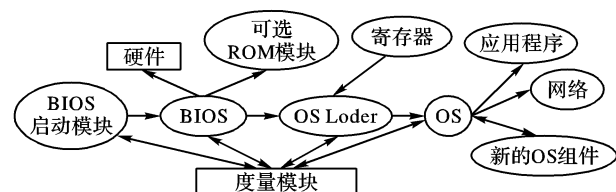


图 2 度量模块的工作原理

2 基于 WDM 的度量模块的设计与实现

WDM(Windows Driver Model)是微软公司提出的全新驱动程序模型,支持即插即用、电源管理和 WMI 技术。在 Windows 操作系统之中,驱动程序也被划分为用户态和核心态,用户态的驱动程序运行在非特权模式上,其他一些被保护的代码也运行在该模式上。用户态的驱动程序不能获得

系统数据的存取权,只有调用 Win32 API 或者系统服务;核心态驱动程序作为操作系统的一个组成部分被执行,支持一个或多个受保护的子系统的操作系统底层组件。用户态和核心态驱动程序有不同的组织结构、不同的入口点和不同的系统接口。一个设备是需要一个用户态驱动程序还是一个核心态驱动程序依赖于该设备的类型和操作系统对它提供的支持。一些设备驱动程序可以完全或部分地运行在用户态。用户态驱动程序没有堆栈空间的限制,可以访问 Win32 API,并可调试。操作系统用设备对象来表示设备。每一个设备都有一个或多个设备对象与之相关联。设备对象提供并封装了在设备上的所有操作。在核心态驱动程序中,每个设备必须至少创建一个设备对象,每一个设备对象在驱动程序堆栈中对应一个驱动程序来管理设备的 I/O 请求。一个设备的所有设备对象被组织成一个设备堆栈。无论何时,一个操作都在一个设备上完成,系统把 IRP(I/O Request Packet)数据结构体传递给设备堆栈中顶部设备的驱动程序。每一个驱动或者处理 IRP,或者把它传递给设备堆栈中下一个设备对象的驱动程序。设备对象由 DEVICE_OBJECT 结构体表示,由对象管理器管理。对象管理器为设备对象提供与其他系统对象相同的功能。特别地,一个设备对象可以被命名,并且一个被命名的设备对象可以有一个句柄。系统为每一个设备对象提供专门的存储空间,被称作设备扩展,设备扩展同设备对象一起被创建和释放。

2.1 硬件固件在系统中的映射原理

在 32 位的 Windows 平台上,系统支持的虚拟地址空间的大小是 4GB(232Byte),而物理地址的大小是随着系统配置而变化的。中央处理器通过使用页表把虚拟地址空间转换到物理地址空间,页表的每一项对应着虚拟地址空间到物理地址空间的映射。硬件抽象层(HAL)使用映射寄存器转换设备地址空间到物理地址空间。由于设备地址空间是独立寻址的,因此设备驱动程序并不能使用指标来访问定位在物理内存中的虚拟地址空间,必须经过映射才能访问。

内存管理器创建和维护了系统的页表,页表入口是一个包含了虚拟地址空间到物理地址空间的转换的系统结构。x86 系统的 32 位的虚拟地址用三部分描述这种映射关系:页目录索引、页表索引和位索引。页目录索引用来定位页表的虚拟地址,而页表索引用来定位页表入口地址。以主板 BIOS 芯片的固件为例,系统默认的最大内存逻辑寻址空间是 00000000h~07FFFFFFh,2GB 以上的内存都保留给系统。在实模式下写地址 0F000:0000h~0F000:FFFFh 时,实际会写到 4GB 的边界,也就是 FFFF0000h~FFFFFFFh 的物理内存,即主板上 Flash EPROM 的芯片的实体译码地址,这就为读写硬件固件提供了手段。Flash EPROM 芯片的译码、存取地址均是从虚拟地址 4GB 的边界往下扩展,接着是可编程控制器(Advanced Programmable Interrupt Controller, APIC)对应的内存地址,然后是 PCI 总线的 I/O 地址。同理,系统中对应其他硬件的固件也有其对应的内存映射地址,原理和主板固件情况类似。表 1 列出了实验环境下主要固件的映射范围。

表 1 主要固件的映射范围

固件名称	映射范围(十六进制)
Broadcom 网卡	E2000000 ~ E200FFFF
Intel 82945 集成显卡	E2100000 ~ E217FFFF
IEEE1394 控制器	E1004000 ~ E10047FF
声卡 UAA 驱动程序	E21C0000 ~ E21C3FFF

2.2 关键数据结构和伪码

随着系统安全性的不断提高,为了使用户态程序能访问

到系统高端地址,必须编写核心态的程序并且进行调试。本文使用 Driver Studio 和 Windows Driver Development Kits (DDK) 开发工具,通过编写 WDM 驱动程序,使用普通用户态的服务控制器(Service Control Manager)调用编写的内核程序计算出硬件固件在虚拟内存中的映射地址。下面以 BIOS 芯片为例,给出了关键的映射过程和伪代码。

1) 映射高端地址到用户态程序空间的伪代码:

```
Switch ( IrpStack -> Parameters. DeviceIoControl. IoControlCode)
{ case IOCTL_READ_IO_MEMORY:
  pMemory = Irp -> AssociatedIrp. SystemBuffer;
  Address. QuadPart = pMemory -> lAddress;
  if (( pBuffer = ( PCHAR) MmMapIoSpace( Address, pMemory ->
    lLength, MmNonCached)) != NULL)
  { RtlCopyMemory ( pMemory -> Buffer, pBuffer, pMemory ->
    lLength);
    Irp -> IoStatus. Information = sizeof ( READ_IO_MEMORY);
  }
  MmUnmapIoSpace ( pBuffer, pMemory -> lLength); }
else { Status = STATUS_INVALID_PARAMETER; Irp -> IoStatus.
  Information = 0; }
```

2) 加载 WDM 驱动的伪代码:

```
If (已经运行) 返回成功;
If (SCM 调用失败) 返回失败;
定义控制服务句柄:
  SC_HANDLE SCService = OpenService( SCManager, driverName,
  SERVICE_ALL_ACCESS);
If (加载驱动)
{ 如果驱动已经加载 返回成功;
  否则返回失败;
}
```

2.3 实验结果

在读取主板固件 BIOS 实验中读取的二进制文件与主板厂商提供的相应 BIOS 二进制比对完全相同,验证了硬件固件

在操作系统中的映射地址,完成了固件的“备份”工作。在进行注入和修改之后 BIOS 固件读取实验中,通过进行数字签名比对,发现固件遭到修改的痕迹。结果表明,整个度量模块能够检测出针对硬件固件的恶意代码和注入行为,为进一步的拦截和恢复提供了基础。

3 结语

本文提出了基于度量模块的针对硬件固件入侵检测方法,不仅为类似 CIH 的新型病毒和基于硬件固件的 Root kit 提供了检测手段,而且可以对操作系统启动进行审计,通过对某些功能进行适当的调整以及和数据库技术的结合,可以进一步对整个涉密网内计算机系统硬件固件进行安全审计,从硬件固件层次为整个应用系统的安全加固提供了有力支撑。

参考文献:

- [1] TRUSTED COMPUTING GROUP (TCG). Main specification version 1. 1 [EB/OL]. <https://www.trustedcomputinggroup.org>, 2003.
- [2] REID JF, CAELLI WJ. Trusted computing and operating system architecture [EB/OL]. <http://crpit.com/confpapers/CRPITV44Reid.pdf>, 2004.
- [3] INTEL CORPORATION. IA-32 Intel Architecture Software Developer's Manual Volume 3: System Programming Guide [M]. Intel Corporation, 2006.
- [4] RUSSINOVICH ME, SOLOMON DA. Microsoft Windows Internals Fourth Edition [M]. Microsoft Press, 2006.
- [5] 陈文钦. BIOS 研发技术剖析 [M]. 北京: 清华大学出版社, 2003.
- [6] DARMAWAN MAPPATUTU SALIHUN. Award BIOS reverse Engineering [J/OL]. The CodeBreakers-Journal, 2004, 1(2).
- [7] DARMAWAN MAPPATUTU SALIHUN. Award BIOS code injection [J/OL]. The CodeBreakers-Journal, 2005, 2(1).
- [8] DUNLAP GW, KING ST, CINAR S, *et al.* ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay [A]. Proceedings of 5th OSDI [C]. Boston, MA, 2002.
- [9] FEDOROVA A, TRAUB O. Logging Options in a Virtual Honeynet [EB/OL]. <http://www.eecs.harvard.edu/fedorova/presentations/logging-honeynet.ppt>, 2006.
- [10] BARHAM P, DRAGOVIC B, FRASER K, *et al.* Xen and the art of virtualization [A]. Proceedings of the 19th ACM Symposium on Operating Systems Principles [C]. 2003. 164-177.
- [11] The Xen project [EB/OL]. <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>, 2006.
- [12] VMware, Inc. VMware virtual machine technology [EB/OL]. <http://www.vmware.com>, 2006.
- [13] WHITAKER A, SHAW M, GRIBBLE SD. Denali: Lightweight Virtual Machines for Distributed and Networked Applications [R]. Technical Report 02-02-01, University of Washington, 2002.
- [14] Bach MJ. UNIX 操作系统设计 [M]. 陈葆钰, 王旭, 柳纯录, 等译. 北京: 北京大学出版社, 1989.
- [15] 毛德操, 胡希明. Linux 内核源代码情景分析 [M]. 杭州: 浙江大学出版社, 2001.
- [16] COLLINS JR. Knark: Linux kernel subversion [Z]. Sans Institute IDS FAQ.
- [17] Stealth, Adore rootkit [EB/OL]. <http://stealth.7350.org/rootkits/>, 2006.
- [18] SD, DEVIK. Linux on-the-fly kernel patching without LKM [EB/OL]. www.phrack.org, 2001-12-28/2004-03-11.
- [19] [C]. Chicago, IL, 2004.
- [20] SNARE project [EB/OL]. <http://www.intersectalliance.com/>, 2006.
- [21] The Honeynet Project "Know Your Enemy: Sebek" [EB/OL]. <http://project.honeynet.org/papers/sebek.pdf>, 2003.
- [22] GOLDBERG RP. Survey of virtual Machine Research [J]. IEEE Computer, 1974, 7(6).
- [23] SINGH A. An Introduction to Virtualization [EB/OL]. <http://www.kernelthread.com/publications/virtualization/>, 2006.
- [24] JIANG XX, XU DY, EIGENMANN R. Protection Mechanisms for Application Service Hosting Platforms [A]. Proceedings of IEEE/ACM Int'l Symposium on Cluster Computing and the Grid (CCGrid 2004)

(上接第 2915 页)

能),因此,攻击者很难从互联网上破坏 Domain0 域。Xen 也提供了在 Domain0 域和用户域间的强隔离,即它们有各自隔离的物理内存、硬盘空间和文件系统。因此,即使用户域已经被攻破,要破坏用户域文件系统上的记录文件或数据库是很困难的。

用户域的 XenLinuxU 内核二进制映像文件存放在 Domain0 域的文件系统中,攻击者无法破坏它;如前所述,用户域的根文件系统也以普通文件存放在 Domain0 域的文件系统中,在用户域运行并完成配置后,为其根文件系统在 Domain0 域上作一备份,使得攻击者也无法破坏它。结果,在攻击者破坏正运行的用户域的文件系统后,从 Domain0 域的备份中恢复用户域的新部署就很简单、迅速。

参考文献: