doi :10.3969/j.issn.1671-1122.2009.05.014

EFI 及其安全性分析

刘冬，文伟平

（北京大学软件与微电子学院信息安全系，北京 102600）

摘　要：随着计算机技术的发展，传统PC BIOS已逐渐成为了现代计算机发展的瓶颈。为了解决BIOS的局限性和相关的问题。Intel公司提出了可扩展固件接口（EFI）的规范标准。作为下一代BIOS，EFI为启动操作系统前的程序提供了一个标准环境。EFI规范具有良好的可扩展性，模块化设计和较低的入门门槛等优势，它大大方便了整个硬件工业的创新和发展。文中详细介绍了EFI，指出EFI存在的一些安全问题，指出了实现EFI安全必须考虑的因素。

关键词：EFI；可扩展接口；BIOS；安全性

中图分类号：TP309　　文献标识码：A

EFI and Security Analysis of EFI Framework

LIU Dong, WEN Wei-ping

(Department of Information Security, SSM, Peking University, Beijing 102600*, China)*

Abstract: With the development of the computer technology, the traditional PC BIOS has become a bottleneck in the development of the modern computer. In order to solve the limitations and difficulties of traditional PC BIOS, Intel has proposed Extensible Firmware Interface (EFI) Specifications. As the next generation of BIOS, EFI provides a standard environment for booting an OS. In contrast with BIOS, EFI components are designed based on modules and could be written with C, which enables good extensibility. EFI largely facilitates the whole PC Industry's innovation. This paper gives a brief introduction of EFI and takes a closer look at some security issues in EFI. Moreover, some key factors that have to be taken into account in a secure EFI environment are proposed.

Key words: EFI; Extensible Firmware Interface; BIOS; security

0 引言

传统计算机操作系统启动前的硬件初始化工作及操作系统的引导控制权都是BIOS（Basic Input/output System)来完成的。随着计算机科学的发展，存在了二十多年的BIOS，越来越与其他设备格格不入，逐渐成为了计算机发展的瓶颈，难以适应现在计算机科学技术的发展要求。首先，与其他设备和计算机技术的飞速发展相比，BIOS的发展是缓慢的。尽管BIOS还能基本满足计算机的基本需求，但其已严重影响了计算机的执行速度。作为DOS时代的产物，BIOS工作在16位的“实模式（Real Mode)”状态下，可管理到的最大内存为1MB，而系统启动时各类扩展卡以及整合设备（如网络控制器、音频控制器）都必须被BIOS访问，但它们的ROM容量也都被限制在128KB。显然，BIOS只能为启动操作系统做准备[1]。其次，BIOS使用汇编语言编写的。作为低级语言，汇编语言具有与底层硬件紧密结合、程序执行速度快以及代码简练的优势。但到今天，采用汇编语言为BIOS增加新功能是相当复杂的，程序编写的难度极大。第三，BIOS是硬件与操作系统的接口，可是这个接口没有统一的标准，各个生产商生产的BIOS不同，致使BIOS升级需要很长时间。针对于以上BIOS的弊端，INTEL开发了下一代BIOS技术“EFI（Extensible Firmware Interface，可扩展固件接口)”。

它定义了操作系统与平台固件之间的接口模型。这个模型提供了平台相关信息、启动服务例程以及操作系统运行时服务例程。操作系统装载器与操作系统可通过接口调用这些服务例程。EFI规范是一个公开的纯接口定义，它不依赖于某个特定的BIOS制造商或某个特定的BIOS实现。它仅定义了平台固件必须实现的接口，以及操作系统可能使用的一系列接口与数据结构，其实现的方式与细节均取决于此规范的实现者。EFI规范还定义了固件驱动程序模型，使得所有遵循此模型开发的固件驱动程序能够相互协作。EFI 主要由一系列包含平台相关信息的数据表（Data Tables）和供操作系统引导程序、操作系统调用的启动服务（Boot Service）和运行时服务（Runtime Service）构成。这些部件联合起来为一个操作系统的启动与预启动程序的执行提供了一个标准环境[2]。

由于 EFI 具有良好的前瞻性，2005 年英特尔把 EFI 规范贡献给业界，成立统一的 EFI 论坛，即管理 EFI 规范的非营利性国际组织 UEFI。共同制订适应于各种平台的接口标准，并于2006 年1月发布了 UEFI2.0版本[2]。UEFI的制订建立在 EFI 1.10 的基础上，现在的最新版本为 2.1。UEFI的主要成员有AMD、AMI、Dell、HP、IBM、Insyde、Intel、Microsoft 和 Phoenix 等[3]。

1 EFI 及框架

1.1 EFI系统框架

UEFI 相关参数定义了操作系统与平台固件间可扩展接口，其最大特点是采用模块化设计，基本分为硬件控制和 OS 软件两大模块，前者只要 EFI 版本相同，功能就完全相同，而后者则是给厂商用 C 语言（而非汇编语言）撰写应用功能的开放接口。通过这个标准的开放接口，厂商可以根据需要自行编写各

种功能插件，比如系统备份/还原插件、浏览器插件、防病毒插件等，同时不受容量限制，这就为固件层级的技术创新提供了平台。图1是EFI的系统构架图，描绘了EFI与固件、硬件、OS加载程序与OS的关系。位于底部的是硬件层，与硬件层直接交互的就是EFI的驱动执行环境(Driver Execution Environment)，它包括协议构架(Protocol Architecture)、平台驱动(Platform Driver)、框架驱动(Framework Driver)、以及兼容支持模块(Compatibility Support Module)等模块化组件，再往上才是EFI以及兼容模式的实现[4]。在整个EFI系统中，驱动执行环境处于基础性地位，它也被称为“Pre-EFI-Foundation(预EFI基础)”。在驱动执行环境的各个组件中，负责与硬件直接交互的便是协议架构(Architecture protocol)模块，它具备与硬件直接交流的能力。在实际执行时，EFI对硬件参数的定义都是通过协议架构来传递的，这一点EFI与传统BIOS并无本质的区别。而决定EFI实现功能的是平台驱动和框架驱动，两者共同为EFI的实际执行提供完整的支撑。同时，EFI系统也包含一个兼容支持模块(Compatibility support module，简称CSM)，它可以在16位实模式下启动计算机以及访问扩展设备的ROM(只读存储器)。这样，即便PC平台中的部分硬件没有专门为EFI设计(例如显卡、声卡等)，EFI的兼容支持模块也可以让它们在整个系统中正常工作，从而有效降低从BIOS到EFI过渡的门槛。

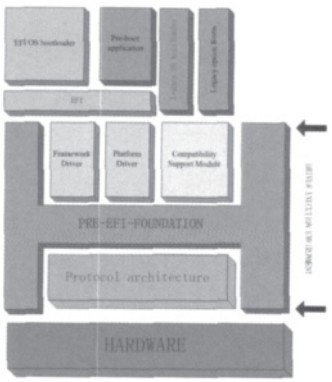

图1 EFI系统构架图

EFI系统的最顶层为“EFI操作系统装载器(EFI OS Loader)”和名为“预启动应用(Pre-boot application)”的扩展工具，EFI系统装载器可以引导操作系统启动，也能够提供一个设定屏幕—EFI同时支持传统的文本界面和图形界面。除了提供基本的硬件参数设定功能外，EFI OS Loader还允许用户设定操作系统的启动顺序(在计算机拥有多个操作系统的条件下)，这相当于直接整合了OS启动管理器；其次，用户也可以通过EFI OS Loader界面来启动扩展工具，例如Ghost系统镜像、磁盘检测、EFI版本升级、病毒查杀或者其他的安全软件等[4]。

1.2 EFI框架流程

EFI通过对特定平台的抽象，提出了一整套数据结构以及接口函数。整个EFI协议几乎是由许多不同的协议组组成。EFI为驱动程序与应用程序的运行准备了一个完全模块化的运行环境，并且这些应用程序与驱动程序是用C语言开发的。开发者可以根据需要对EFI规范中某些相关协议组进行扩充(完成函数体的编写)，实现需要的功能。

EFI开发软件包(EFI Develop Kit，EDK)是英特尔对于EFI规范的实现。EDK以分阶段的方式初始化平台(如图2所示)。EDK启动过程分为4个主要阶段：SEC(Security)阶段、PEI(Pre-EFI Initialization)阶段、DXE(Driver Execution Environment)阶段和BDS(Boot Device Selection)阶段[4]。SEC阶段是电脑加电后的第一个阶段，主要负责验证平台硬件信息。PEI阶段的目的只是为了找到并最少地初始化内存。DXE阶段是大部分系统初始化的阶段，包括DXE核(DXE Core)、DXE分发器(DXE Dispatcher)和一系列DXE驱动。DXE核产生一系列的启动服务(Boot Services)、运行服务(Runtime Services)和DXE服务(DXE Services)。DXE分发器负责发现和以正确的顺序执行DXE驱动。DXE驱动则负责初始化处理器、芯片以及为系统服务、控制设备和启动设备提供软件提取的组件。BDS阶段和驱动执行阶段一起建立控制台，并从启动设备中启动操作系统。EDK启动完成后就进入TSL(Transient System Load)和RT(Run time)阶段。TSL阶段是操作系统加载阶段。当操作系统加载完毕时，进入RT阶段。这时，从DXE阶段开始的大部分服务已终止，对处理器和平台资源的所有权从平台固件变为操作系统。AL(After2life)阶段是平台控制从操作系统返回到平台固件的阶段，是RT阶段的延长部分。AL阶段以系统重启或以操作系统从睡眠状态醒来结束[5]。

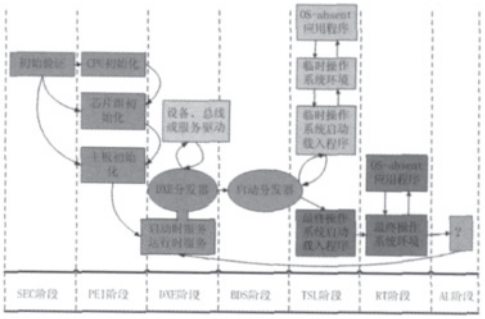

图2 EFI框架流程

2 EFI的安全性问题

EFI BIOS在功能上超越了传统的BIOS，解决了BIOS设置难、扩展难等问题，但由于其组件加载模式、使用C语言编写也带来了许多安全威胁。我们需要更加重视它的安全性。

2.1 缓冲区溢出漏洞

因为EFI是用C语言实现的，在程序中静态变量被分配在数据段中，动态变量分配在堆栈段，它是一个众所周知的特点，就是语法限制不太严格，程序设计自由度大。例如，C语言本身不进行数组的边界的检查，而由程序员编写者保证程序的正确。在动态分配缓冲区后，如果尝试向该缓冲区写入超出其所能装载的数据，就会覆盖掉堆栈的其他数据，甚至覆盖函数的返回地址，这事就发生了缓冲区溢出现象。利用缓冲区溢出进行攻击是黑客常用的攻击手段。堆栈缓冲区

溢出就是典型的攻击形式。而目前的静态和动态检测方法还不能有效地解决缓冲区溢出的问题。EFI 框架中，从 DXE 阶段开始就可以自由分配系统内存，加之 EFI 驱动模式的灵活应用，因此缓冲区溢出漏洞成为 EFI 不可避免的安全隐患。

2.2 EFI安全启动

EFI 的文件系统并不是存储于主板的 ROM 中，EFI 需要更大的存储空间，于是在硬盘隔离出一个小分区作为存储空间，拥有独立的文件系统、独立的控制功能模块。这样虽可装入更多实用的工具，如硬盘分区、多操作系统引导、系统备份和恢复等，甚至能作为数字版权控制工具和电子安全防范工具，但硬盘是相当脆弱的配件之一，出现问题的话后果不堪设想。我们知道，硬盘是整个计算机系统中最“脆弱”的部件，平时频繁的读盘操作，甚至轻轻的撞击，都可能会使硬盘出现物理坏道，甚至寿终正寝。一旦硬盘出现问题，容易使某个功能模块的完整性受到破坏。

2.3 网络安全

EFI 在其所有组件都加载完成后，可以在终端开启一个 SHELL 程序，使得系统在不进入操作系统就可以进行系统配置、诊断等操作，另一方面，EFI 规范提供了 ARP、TCPv4、IPv4 和 UDPv4 等网络协议。这些网络协议的加载和应用，又使得平台可以执行远程启动及配置操作，于是在连接通讯前，服务器和客户端都有必要对对方进行身份鉴别和完整性检测，保护通信及系统平台的安全性。例如，在一个局域网内，所有系统平台都是基于 EFI，服务器需要不定时地统计所有客户机的状态信息，或者发送控制命令（如重启、关机命令）给某客户机。因为 EFI 具有硬件驱动，可以独立地控制系统。所以在使用 EFI 的网络协议实现通讯时，如果缺少相应的安全检测，将会对系统平台的安全构成威胁。

3 EFI 安全服务机制

EFI 提供了相应的机制来保护其框架的安全性。配置安全的 EFI 环境关键在于运行可信程序的保障能力和 EFI 映像的完整性检测能力。EFI 的完整性和鉴别授权验证是基于公钥密码、单向函数、数字签名、公钥证书和可信根等密码体制。EFI 框架在 PEI 向 DXE 阶段转换的过程中，DXE 会加载一系列安全服务，其中包括[6]：

- 支持启动完整性服务BIS(Boot Integrity Services)和可信计算组织 TCG 等标准的安全服务；
- 安全体系架构协议；
- 映像的签名实现。

EFI BIS 协议提供了校验 EFI 映像完整性的有效策略机制。它引入了启动授权 BA 公钥(Boot Authentication Public Key) 的概念。平台系统的合法所有者拥有一对 BA 公、私钥，用来实施 EFI 驱动及应用程序在平台上运行的授权。映像加载过程中，系统会对每个驱动或应用程序进行完整性检测，判断其是否得到合法授权。与此同时，系统会对目标对象的附属签名清单（该清单包括每个对象相关属性、HASH 值及其数字签名等）重新签名。之后，系统使用 BA 公钥对所有的模块或应用程序及清单进行签名验证，从而保证平台的安全性。这些服务为平台提供了多种安全功能，如密码原函数、获取及更新 BA 公私钥对、验证对象清单和记录验证结果并执行应用程序等。安全体系架构协议和 BDS 启动管理器使用这些安全服务来查找、加载和执行平台驱动。平台启动中所有发现的模块和驱动的完整性及真实性也都会通过这些安全服务来实现。

4 结论

为解决传统 BIOS 的局限性而提出了 EFI BIOS，它是一个全新的预启动平台。在这个平台上进行网络接入认证的研究是一个新颖、有创意和挑战性的工作。EFI 系统上进行的应用开发与在其他操作系统上的不同，它的开发难度更大且在 EFI 系统上的研究工作缺少前人的经验，目前正处于不断的尝试和摸索中。

与传统 BIOS 相比较，EFI 体现出了更多的优势及发展空间。当 EFI 的应用日益广泛时，做好 EFI 及其框架的安全性研究是十分必要的。实现安全的 EFI 环境关键在于运行安全可信的程序代码，并且检测 EFI 映像文件的完整性，以确保其未被非法篡改。

综上所述，BIOS 作为 PC 体系中的一个关键组成，不论基于经济因素，安全因素，还是发展因素等，都有必要研制中国自己的 PC BIOS，并在此基础上，发展国产操作系统，即可构筑真正令人放心的国产系统软件平台。近年来，Intel 推出了 EFI 接口标准，目前正值传统 BIOS 向新一代 BIOS 规范的过渡阶段，是一个发展自主知识产权的国产 BIOS 技术与产品的极好时机。 （责编 张岩）

参考文献：
[1] 吴丽军.“计算机 BIOS 安全检查技术的研究与实现”[D]. 西安电子科技大学军事学专业硕士生毕业论文，2008,1: 1-26.
[2] Intel Corporation. Extensible Firmware Interface Specification Version 1.10. http://developer.intel.com/technology/efi/, 2002.12.
[3] Unified EFI Forum. Unified Extensible Firmware Interface Specification Version 2.0. http://www.uefi.org/specs/, 2006.01.
[4] Intel Corporation, "EFI 1.10 Driver Writer's Guide Version 0.7", Jun. 2003. http://developer.intel.com/technology/efi/.
[5] Intel Corporation. Intel Platform Innovation Framework for EFI Architecture Specification[S].Version 0.9, 2003,09.
[6] 谢勇. 基于EFI双核的安全系统框架的设计与研究[D]. 上海交通大学计算机体系结构专业硕士生论文，2008,1：1-19.

作者简介：刘冬（1983-），男，硕士研究生，主要研究方向：网络安全；文伟平（1976-），男，副教授，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等。