

BIOS 陷门实现机理及检测技术研究

姜子峰*, 曾光裕, 王 炜, 高洪博

(信息工程大学, 郑州 450002)

(*通信作者电子邮箱 jzf283600@163.com)

摘 要:基本输入输出系统(BIOS)陷门对计算机系统影响巨大,且现有工具难以有效检测其存在。在逆向分析基础上,研究了 BIOS 结构及 BIOS 代码混淆技术。根据实现粒度,将 BIOS 陷门分为模块级 BIOS 陷门与指令级 BIOS 陷门,详细分析了这两类陷门的实现原理与特点。最后提出了基于模块构成分析的模块级陷门检测方法和基于完整性度量的指令级陷门检测方法。实验结果表明,两种方法能有效检测与之对应的 BIOS 陷门的存在。

关键词:BIOS 陷门;逆向分析;代码混淆;模块级陷门;指令级陷门;陷门检测

中图分类号: TP309 **文献标志码:** A

Research on implementation mechanism and detection technique of BIOS trapdoor

JIANG Zifeng*, ZENG Guangyu, WANG Wei, GAO Hongbo

(Information Engineering University, Zhengzhou Henan 450002, China)

Abstract: Basic Input Output System (BIOS) trapdoor has huge impact on computer system, and it is difficult to detect the existence of BIOS trapdoor effectively with the existing tools. After researching BIOS structure and BIOS code obfuscation technique based on reverse analysis, BIOS trapdoors were divided into module-level BIOS trapdoor and instruction-level BIOS trapdoor according to implementation granularity, followed by analyzing the implementation principle and characteristics of these two BIOS trapdoors in detail. Finally the detection method of module-level trapdoor based on analyzing module structure and the detection method of instruction-level trapdoor based on integrity measurement were presented. The experimental results show that these two methods can detect the existence of their corresponding BIOS trapdoors effectively.

Key words: BIOS trapdoor; reverse analysis; code obfuscation; module-level trapdoor; instruction-level trapdoor; trapdoor detection

0 引言

大量的信息安全事件使人们逐渐认识到,大多数安全隐患来自于计算机终端,因此必须确保微机终端的信息安全^[1]。近年来,针对微机终端的攻击已经由操作系统深入到了系统的底层,如基本输入输出系统(Basic Input Output System, BIOS)。BIOS 存储在微机主板上的非易失存储器中,加电后首先执行,提供最基本的硬件初始化、外围控制等功能,并引导操作系统启动。随着计算机技术的发展和应用需求的变化,存储 BIOS 的介质由 ROM 芯片变为容量更大的支持纯软件方式读写的 FLASH 芯片^[2],这使得第三方向 BIOS 中植入特殊逻辑更为便捷,BIOS 安全问题逐渐显现。1998 年,CIH 病毒通过攻击计算机 BIOS 等手段,使得全球数百万台计算机遭到破坏^[3]。2006 年,在“Black Hat”会议上,Heasman^[4-5]阐述了一种新的 Rootkit 技术,利用该技术在计算机 BIOS 中隐藏 Rootkit 恶意代码并使之在操作系统运行过程中生效。2009 年,Ortega 等^[6]将一小段代码植入 BIOS 获得整台计算机的控制权,即使重启系统甚至刷新 BIOS 都无法清除。

综上所述,通过向 BIOS 中嵌入特殊逻辑形成 BIOS 陷门,可实现包括功能破坏、信息获取与状态监控在内的特殊功能。BIOS 先于操作系统运行,可在主流安全软件运行前加载,导致传统方法难以检测 BIOS 陷门。此外,针对 BIOS 陷门检测

方面的研究还相对滞后,并没有一种统一的公认有效的检测方法。为达到检测 BIOS 陷门的目的,必须首先分析并掌握 BIOS 陷门的实现原理。本文在分析 BIOS 结构及代码混淆技术的基础上,重点研究 BIOS 陷门实现原理,最后提出 BIOS 陷门检测方法。

1 BIOS 结构分析

BIOS 保存在可读写的 FLASH 芯片中,利用官方提供的工具能够从芯片中提取出 BIOS 的二进制映像文件,再利用 IDA 等工具对该文件进行逆向分析可得到该 BIOS 的内部结构及相关信息。表 1 给出了某 Award BIOS 的结构分析结果。

表 1 Award BIOS 主要结构

Size/kB	模块名称	功能简介	是否压缩
74.77	System BIOS	核心模块	是
30.46	XGROUP CODE	设备初始化等	是
5.55	ACPI table	ACPI 表	是
15.42	YGROUP ROM	厂家扩展功能	是
6.99	GROUP ROM[0]	ASCII 字符串	是
31.42	PCI ROM[A]	驱动程序等	是
290.54	空余区域	以 FFH 填充	
3.19	Decompression block	解压缩模块	否
8	Boot block	基础启动模块	否

收稿日期:2012-08-20;修回日期:2012-10-25。 基金项目:信息工程大学未来发展基金资助项目(1201)。

作者简介:姜子峰(1988-),男,河北石家庄人,硕士研究生,主要研究方向:信息安全; 曾光裕(1966-),女,重庆人,副教授,硕士,主要研究方向:信息安全; 王炜(1975-),男,湖北武汉人,讲师,博士,主要研究方向:信息安全、计算机体系结构; 高洪博(1984-),男,河北景县人,博士研究生,主要研究方向:信息安全。

分析发现,该 BIOS 采用线性排列的模块化构成方式,由两个未压缩模块、若干个压缩模块以及用 FFH 填充的空余区域组成。其中,Boot block 模块是 BIOS 最先执行的模块,计算机加电之后会运行 F000:F00h 处的一条跳转指令^[7],该指令会指向 Boot block 模块,使 BIOS 得以运行。BIOS 的模块在压缩后,会在头部标明相关信息(如模块名称、校验和、大小、压缩算法等),而各压缩模块通过解压缩模块释放到指定内存后方可运行。此外,BIOS 文件中通常会有大小不一的以 FFH 填充的空余区域,这就为在 BIOS 中嵌入特殊逻辑提供了可能。

2 BIOS 代码混淆技术

通过对可执行二进制代码进行逆向分析,攻击者可以获得代码内部细节,进而对代码进行篡改或者窃取其中的商业机密。为防范这种情况的发生,BIOS 厂商在 BIOS 制造过程中融入了代码混淆技术。无论要实现某种 BIOS 陷门,或者针对某种 BIOS 陷门进行检测,在没有 BIOS 源代码的前提下,都必须针对 BIOS 二进制代码进行逆向分析,掌握 BIOS 的内部逻辑及功能,而要想得到正确的逆向分析结果,必须克服 BIOS 代码混淆技术所造成的负面影响。BIOS 中常见的混淆技术主要有如下几种。

1) 混淆数据和指令。通过把数据嵌入到代码段区域,使得反汇编者在无法正确区分数据和代码的情况下,得出错误的反汇编结果。借助 IDA 得到的正确的反汇编结果如下所示:

```
E1D3      jmp      near ptr loc_E6A6 + 1
E1D3; -----
E1D6      dw 0E1D8h
E1D8; -----
E1D8      mov     ax, 0F000h
E1DB      mov     ds, ax
```

在正确的结果中,偏移地址 E1D6 处存放数据 0E1D8h,而在其周围都是指令,若误把 0E1D8h 当作指令处理,则会得出错误结果,如下列反汇编结果所示:

```
E1D3      jmp      near ptr loc_E6A6 + 1
E1D3; -----
E1D6      db 0D8h; ?
E1D7; -----
E1D7      loope   near ptr loc_E18F + 2
E1D9      add     al, dh
E1DB      mov     ds, ax
```

2) 插入冗余代码。在不影响正常代码运行的情况下,通过插入不会执行的代码,影响反汇编结果的正确性。如下列反汇编结果所示,偏移地址 E097 处存放了一条不会执行的跳转指令,若沿着该指令继续反汇编,则会得到错误结果,影响后续分析。

```
E095      jnz     short loc_E09A
E097      jmp     near ptr loc_F619 + 1
```

3) 多跳转混淆。在不明显增加 BIOS 运行时间开销的情况下,通过大量使用段间及段内跳转指令,增加 BIOS 代码的反汇编难度。此外,如下列反汇编结果所示,即使一定范围内能够正确得到反汇编结果,大量的跳转指令也使得代码的可阅读性与可修改性变差,增加分析难度。

```
E1E9      out     dx, al
E1EA      jmp     short loc_E1EC
E1EC; -----
E1EC      jmp     short loc_E1EE
```

```
E1EE; -----
E1EE      loop    near ptr loc_E1E4 + 1
```

4) 目标地址隐藏。通过将 call、jmp 等显式影响目标地址的指令替换为隐式影响目标地址的指令(如 ret 等),从而将代码的控制流变得复杂,增加反汇编难度,达到隐藏原始跳转目标地址的目的。如下列反汇编结果所示,通过对堆栈指针 SP 进行赋值,然后利用段内返回指令 retn 对指令指针 IP 进行操作的特点^[8],将真正要执行的位于偏移地址 E073 处的跳转指令隐藏起来。该混淆技术在 BIOS 中大量应用。

```
E06B      mov     sp, 0E071h
E06E      jmp     loc_F693
E06E; -----
E071      dw      0E073h
E073; -----
E073      jnz     loc_E09A
F693      xor     al, al
F695      retn
```

5) 条件跳转混淆。通过人为构造条件跳转指令,添加干扰跳转分支,并控制寄存器的值来控制跳转结果,增加阅读与分析的难度,达到混淆原始跳转目标地址的目的。如下列反汇编结果所示,通过将 jmp 指令替换为由 cmp、jz 和 jnz 组合而成的代码段,通过控制寄存器 al 的值来控制跳转结果,隐藏原始的跳转目标地址。

```
E08D      cmp     al, 0FFh
E08F      jz      short loc_E09A
E091      and     al, 1Ch
E093      cmp     al, 0Ch
E095      jnz     short loc_E09A
E097      jmp     near ptr loc_F619 + 1
```

BIOS 代码混淆技术是 BIOS 陷门攻防的关键环节。一方面,该技术既可以增加针对 BIOS 二进制文件的逆向分析难度,又可以增加 BIOS 代码的耦合度,使得直接修改 BIOS 代码来实现 BIOS 陷门变得十分困难;另一方面,一旦成功攻克代码混淆技术,嵌入了 BIOS 陷门,该陷门就可以利用代码混淆技术来隐藏自己,使得相应的检测及清除难度大大增加,除非进行 BIOS 二进制文件的完整性度量^[9],否则很难发现 BIOS 是否已经被修改。通过对 BIOS 代码混淆技术进行研究,有利于 BIOS 二进制文件的逆向分析,为进一步实现 BIOS 陷门的嵌入和检测提供必要的技术支持。

3 BIOS 陷门

BIOS 存储在可读写的 FLASH 芯片内,采用模块压缩化的结构,各模块之间线性排列,且内部通常包含一定容量的空余区域。这种设计思想一方面便于 BIOS 升级和维护,另一方面也为 BIOS 陷门的嵌入提供了可能。根据实现粒度,本文将 BIOS 陷门分为模块级陷门和指令级陷门两类。

3.1 模块级陷门

模块级陷门是指通过构造 BIOS 可执行模块,进而嵌入 BIOS 以实现特殊逻辑的 BIOS 陷门。BIOS 可执行模块主要有两种,分别是工业标准架构(Industry Standard Architecture, ISA)模块和外围部件互连(Peripheral Component Interconnect, PCI)模块。

1) ISA 模块陷门。ISA 总线是 PC/AT 微机上的系统总线,后被 PCI 总线替代。ISA 模块一般用于管理集成的 ISA 控制器,现在的主板基本没有 ISA 控制器,但出于兼容性考虑,BIOS 中仍然保留了 ISA 模块。ISA 模块的结构如表 2 所示。

利用 ISA 模块嵌入 BIOS 陷门并不复杂,需要将陷门放入模块主体,按照 ISA 模块结构对陷门进行封装,以字节为单位计算模块的校验和并存放于模块结尾处,按照指定压缩算法进行压缩,在 BIOS 文件中搜寻一块范围大于该 ISA 模块的空余区域,将该 ISA 模块覆盖此空余区域,最后计算并写入 BIOS 校验和即可。

表2 ISA 模块结构

区域	偏移	值	功能
模块头部	00H	AA55	识别标志
	02H	N	模块大小为 $N * 512$ B
	03H	X	模块主体入口点 X
模块主体	XH		
模块结尾	模块单字节校验和		

2)PCI 模块陷门。PCI 总线是当前微机中常用的总线之一,具有性能好、兼容性强等诸多优点。PCI 总线规范提供了一种机制,允许 PCI 设备携带一个与系统内存空间无关的扩展 ROM,在其中存放 PCI 模块,用于设备的初始化和系统的启动等工作。此外,主板厂商为使主板上的 PCI 设备能够正常工作,也会在 BIOS 内部嵌入相应的 PCI 模块。PCI 模块的结构如表 3 所示

表3 PCI 模块结构

区域	偏移	值	功能
模块头部	00H	AA55	识别标志
	02H	N	模块大小为 $N * 512$ B
	03H	X	模块主体入口点 X
	06H		保留
	18H	Y	PCI 数据结构的偏移值
PCI 数据结构	YH		
模块主体	XH		

利用 PCI 模块嵌入陷门的方法与利用 ISA 模块嵌入陷门的过程相似(如图 1 所示),但前者实现更复杂。PCI 模块需要构造正确的 PCI 数据结构,该数据结构包含供应商识别码和设备识别码等信息,这两个识别码必须属于真实的设备,否则 PCI 模块无法执行,因此需要伪造一个 PCI 数据结构,使得 PCI 模块内的供应商识别码与设备识别码同真实 PCI 设备相一致。另一方面,相对于 ISA 模块来说,由于 PCI 设备种类繁多,所以利用 PCI 模块可以实现功能更加复杂的 BIOS 陷门,其实用价值更大^[10]。

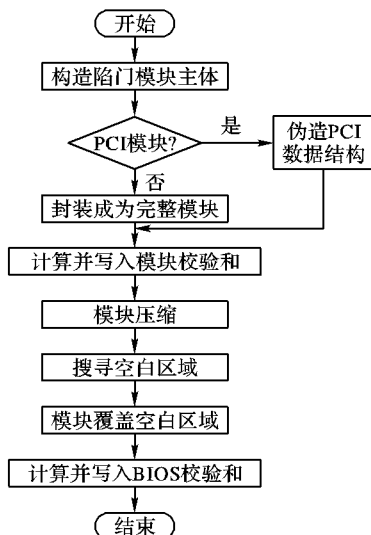


图1 模块级陷门嵌入流程

3.2 指令级陷门

指令级陷门是指通过改变已有 BIOS 模块内部的代码执行流程,使得嵌入 BIOS 中的特殊逻辑代码在 BIOS 运行过程中得以执行的 BIOS 陷门。

BIOS 中采用了多种代码混淆技术,因此很难通过直接修改 BIOS 二进制文件中的原始代码区数据来嵌入陷门,即使嵌入成功,也会改变原始 BIOS 功能,易被用户察觉。本文借鉴 HOOK 思想,在不改变原始 BIOS 功能的前提下,通过修改目标地址来嵌入指令级陷门,其过程主要分为三个阶段:首先,在一定范围内正确反汇编 BIOS 二进制文件的基础上,将自定义的 BIOS 陷门嵌入某一 BIOS 模块内部,该模块必须包含足够大的能容纳 BIOS 陷门代码的空余区域;其次,修改该模块中的某一跳转指令,使目标地址指向 BIOS 陷门入口,修改陷门返回地址,保证 BIOS 陷门代码结束后,正确返回 BIOS 原始目标地址处继续执行;最后,修改该模块校验和,若被修改的模块是压缩模块,则还要涉及到模块解压缩、模块提取及模块嵌入等操作。指令级陷门的嵌入流程如图 2 所示。

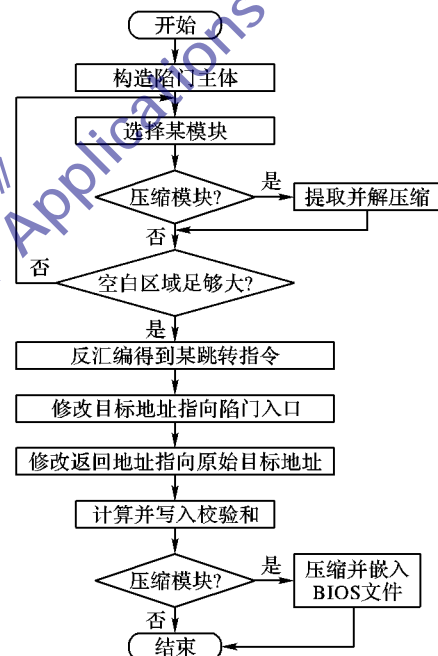


图2 指令级陷门嵌入流程

对于本文所讨论的两种 BIOS 陷门来说,由于 BIOS 内部各模块在运行过程中所拥有的硬件资源不一致,所以依托不同模块进行嵌入的陷门所能实现的功能就会有所差异。例如,Boot block 模块在加电后首先执行,而在这个阶段诸如内存等硬件还没有初始化,导致很难实现较复杂的陷门;ISA/PCI 模块在大部分硬件初始化之后运行,因此可以实现较为复杂的软硬件深度结合的陷门,但由于该阶段内硬盘还没有初始化,导致无法直接利用 13 号中断进行硬盘操作。

4 BIOS 陷门检测

模块级陷门与指令级陷门具有不同特点,因此需要根据其自身特点分开讨论与之对应的陷门检测方法。

4.1 模块级陷门检测

要实现模块级陷门,需要将相关内容伪装成标准 BIOS 模块,如 ISA 模块和 PCI 模块。但模块级陷门嵌入 BIOS 原始文件之后,会对原始文件的模块构成产生影响,模块的排列方式和模块数量都会发生变化,因此本文提出基于模块构成分析的模块级陷门检测方法,通过和原始 BIOS 文件模块构成进行

对比,可判断 BIOS 文件是否被嵌入模块级陷门,具体的检测方法如图 3 所示。

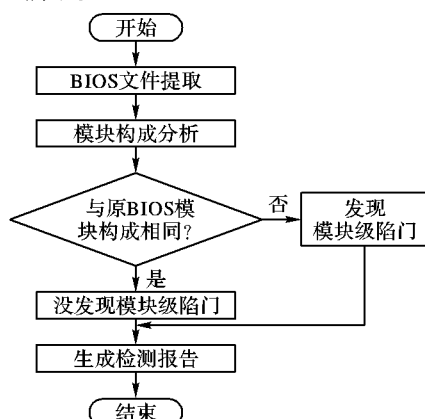


图3 基于模块构成分析的模块级陷门检测方法

该检测方法从 FLASH 芯片中提取 BIOS 文件,并对其内部模块构成进行分析,获得模块的排列方式和数量等信息。然后将该 BIOS 的模块构成同原 BIOS 进行对比,若相同,则说明没有发现模块级陷门;反之,则发现模块级陷门。该方法需要对比原始 BIOS 的模块构成来判断现有 BIOS 内部模块是否发生改变,因此该检测方法的应用前提是能够得到原始 BIOS 文件。

4.2 指令级陷门检测

指令级陷门的实现需要依托于 BIOS 文件中的已有模块。鉴于不同版本的 BIOS 之间以及同一 BIOS 的不同模块之间有着不同的代码校验和生成算法,如 ISA/PCI 模块以字节为单位计算校验和,而 Boot block 是以字为单位计算校验和,因此并没有一种统一的指令级陷门的实现方法;另一方面,也正是由于这种非统一性以及 BIOS 二进制文件较高的代码耦合度与较差的可阅读性,再加上指令级陷门的实现粒度更小,使得指令级陷门具有比模块级陷门更高的隐蔽性。

针对指令级陷门的特点,提出基于完整性度量的指令级陷门检测方法。无论指令级陷门依托何种模块进行嵌入,都要对该模块进行一定程度上的修改,破坏模块的完整性,因此采用完整性度量方法可发现指令级陷门的存在,并能够定位陷门所在模块,具体的检测方法如图 4 所示。

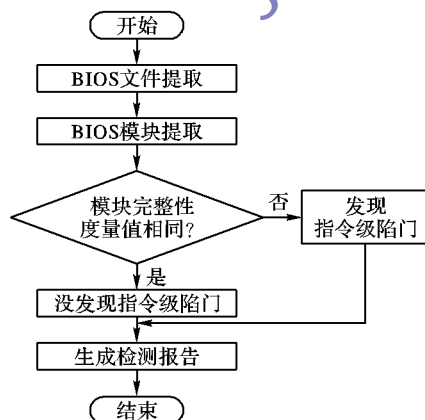


图4 基于完整性度量的指令级陷门检测方法

该检测方法从 FLASH 芯片中提取 BIOS 文件,并从中提取出所有内部模块,依次对这些模块进行完整性度量,即与原始 BIOS 模块的完整性度量值进行对比,若二者数值相同,则说明没有发现指令级陷门;反之,则发现指令级陷门。该方法需要对比原始 BIOS 内部模块的完整性度量值来判断现有 BIOS 内部模块的完整性,因此该检测方法的应用前提是能够

得到原始 BIOS 文件。

4.3 实验结果

为便于研究,本文设计实现了两种 BIOS 陷门作为检测实验样例。

1) 模块级陷门。19 号中断的调用是 BIOS 将控制权转移给操作系统加载器的必经环节,因此通过挂钩 19 号中断,可以劫持系统的控制权,从而控制启动流程,实现特殊功能,这正是许多 Rootkit 的实现原理。本文利用 19 号中断,通过构造 ISA 模块,实现了一个模块级陷门,该陷门具有如下功能:通过改变中断向量表中原始 19 号中断向量的值来控制 BIOS 的正常运行,执行陷门代码,实现特殊功能,显示特定字符串,在特殊功能实现完成后,恢复原始 19 号中断向量的值,跳转到原始目标地址处,使 BIOS 继续正常引导操作系统。如果将显示字符串的功能替换成其他功能,就能够实现更为复杂的陷门,例如替换成挂钩 13 号中断,就能实现磁盘读写功能;替换成远程通信功能,就能借助于网卡实现更复杂的远程传输与控制功能等。

2) 指令级陷门。利用特殊手段触发隐藏在系统内部的特殊逻辑,能进一步提高陷门的隐蔽性。本文采用了特殊逻辑触发机制,通过在已有 PCI 模块内修改和嵌入代码,实现了一个指令级陷门,该陷门具有如下功能:在 BIOS 正常运行过程中,提示用户输入身份认证密码,然后利用键盘输入与时间定时两种手段,可触发嵌入到 BIOS 模块内的指令级陷门,实现特殊功能,显示特定字符串,并中断 BIOS 的正常运行;在特殊功能实现完成后,再通过输入特定指令来恢复 BIOS 的正常运行。BIOS 自带的密码存储于 CMOS 中,通过放电可以使 BIOS 自带密码失效^[11],而本文实现的身份验证功能存储于 FLASH 芯片内,除非刷新 BIOS,否则无法绕过该验证功能;大部分情况下,BIOS 可以正常工作,一旦通过特定指令输入或者时间定时触发了嵌入在 BIOS 中的陷门,就可以劫持系统的启动过程,从而实现特定功能。此外,借助于诸如温度传感器等硬件设备,能够实现诸如温度触发等更为隐蔽的触发手段。

Bochs 是一个具有可移植性的 x86 平台开源模拟器,可以在大部分主流平台上运行,包括对 Intel x86 CPU、通用 I/O 设备以及可定制 BIOS 的模拟^[12]。本文利用 Bochs 对实现的 BIOS 陷门进行功能测试。测试结果如图 5 所示。

如图 5(a)所示,该模块级陷门通过挂钩 19 号中断获得系统控制权,进而完成特定功能,最后将控制权交还给 BIOS,使系统继续正常运行;图 5(b)所示的指令级陷门成功实现了密码认证功能,并且可以在 BIOS 正常运行过程中通过键盘输入和时间节点两种方式隐秘触发特定功能,最后恢复到系统正常运行状态。以上 BIOS 陷门的测试结果表明,本文所设计的两个 BIOS 陷门均能发挥既定功能,影响原始 BIOS 正常工作,对系统造成危害。

3) BIOS 陷门检测。本文利用常用 BIOS 分析工具 CBROM 和 Award BIOS Editor 以及本文所提 BIOS 陷门检测方法对嵌入两种 BIOS 陷门的 BIOS 进行检测,检测结果如表 4 所示。

表4 BIOS 陷门检测结果

检测对象	CBROM	Award BIOS Editor	本文检测方法
模块级陷门	是	是	是
指令级陷门	否	否	是

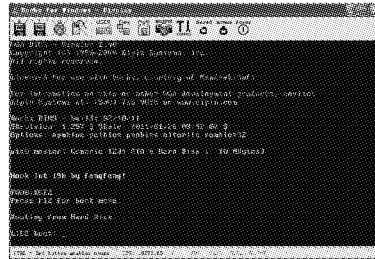
测试结果表明:

1) 模块级陷门无法躲避 BIOS 分析工具和本文检测方法

的检测。这是因为模块级陷门以整个模块为单位,按照 BIOS 模块的标准格式进行嵌入,改变了 BIOS 文件的模块构成,因此,利用基于模块构成分析的 BIOS 分析工具和本文检测方法都可以发现模块级陷门的存在。

2) 指令级陷门能够躲避 BIOS 分析工具的检测,却无法躲避本文所提方法的检测。其原因是指令级陷门在 BIOS 现有模块的内部进行修改,并没有改变 BIOS 文件的模块构成,具有更

高的隐蔽性,因此能够躲避基于模块结构分析的 BIOS 分析工具的检测。然而,无论指令级陷门如何实现,都会对原始 BIOS 文件进行修改,同样会改变 BIOS 文件的完整性度量值,因此利用本文所提出的基于完整性度量的检测方法,可以发现指令级陷门的存在。指令级陷门完整性度量结果如表 5 所示,除 PCI ROM[A] 模块外,其余模块的完整性度量值都没有变化,因此可以判断,PCI ROM[A] 模块中嵌入了指令级陷门。



(a) 模块级陷门



(b) 指令级陷门

图5 BIOS 陷门测试结果

表5 指令级陷门完整性度量结果

模块名称	完整性度量值	
	原始 BIOS 文件	嵌入指令集陷门的 BIOS 文件
System BIOS	38D450A7	38D450A7
XGROUP CODE	541911B6	541911B6
ACPI table	696EE330	696EE330
YGROUP ROM	59F1B1D0	59F1B1D0
GROUP ROM[0]	31303AC3	31303AC3
PCI ROM[A]	55A08004	6EB5D648
Decompression block	94146927	94146927
Boot block	EF63726E	EF63726E

5 结语

随着计算机技术的发展,BIOS 安全的重要性日益凸显。目前,针对 BIOS 的攻击逐渐增多,而 BIOS 安全防护研究还相对滞后。本文在分析 BIOS 内部结构及代码混淆技术的基础上,研究了 BIOS 陷门的实现原理,提出了 BIOS 陷门检测方法,通过实验验证了检测方法的有效性。下一步的工作重点是在无法获得原始 BIOS 的情况下,研究如何有效地检测 BIOS 陷门。

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 E 辑:信息科学,2007,37(2):129-150.

- [2] ZIMMER V, ROTHMAN M, HALE R. Beyond BIOS: implementing the unified extensible firmware interface with Intel's framework [M]. [S. l.]: Intel Press, 2006: 2-9.
- [3] 池亚平,许盛伟,方勇. BIOS 木马机理分析与防护[J]. 计算机工程, 2011, 37(13): 122-124.
- [4] HEASMAN J. Implementing and detecting an ACPI BIOS rootkit [EB/OL]. [2012-07-10]. <http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Heasman.pdf>.
- [5] HEASMAN J. Implementing and detecting a PCI rootkit [EB/OL]. [2012-07-10]. <http://www.blackhat.com/presentations/bh-de-07/Heasman/Paper/bh-de-07-Heasman-WP.pdf>.
- [6] ORTEGA A, SACCO A. Persistent BIOS infection [EB/OL]. [2012-07-10]. <http://www.coresecurity.com/content/Persistent-Bios-Infection>.
- [7] KASPERSKY K. Shellcoder 编程揭秘[M]. 罗爱国,郑艳杰,译. 北京:电子工业出版社,2006:348-376.
- [8] 王爽. 汇编语言[M]. 北京:清华大学出版社,2008.
- [9] 周振柳,刘宝旭,池亚平,等. 计算机 BIOS 安全风险分析与检测系统研究[J]. 计算机工程, 2007, 33(16): 114-116.
- [10] 王晓霞,刘宝旭,潘林. BIOS 恶意代码实现及其检测系统设计[J]. 计算机工程, 2010, 36(21): 17-21.
- [11] Removing a Bios-CMOS password [EB/OL]. [2012-07-10]. http://www.dewassoc.com/support/bios/bios_password.htm.
- [12] Introduction to Bochs [EB/OL]. [2012-07-10]. <http://bochs.sourceforge.net/doc/docbook/user/introduction.html>.

(上接第 454 页)

- [7] FAN C-P, HWANG J-K. Implementations of high throughput sequential and fully pipelined AES processors on FPGA [C]// IS-PACS 2007: Proceeding of 2007 International Symposium on International Signal Processing and Symposium and Communication Systems. Piscataway: IEEE, 2007: 353-356.
- [8] SEVER R, ISMAILGLU A N, TEKMEYEN Y C, et al. A high speed FPGA implementation of the Rijndael algorithm [C]// DSD 2004: Euromicro Symposium on Digital System Design. Piscataway: IEEE, 2004: 358-362.
- [9] CHEN R J, PENG Y C, LAI J L, et al. Architecture design of high efficient and non-memory AES crypto core for WPAN [C]// NSS '09: Third International Conference on Network and System Security. Piscataway: IEEE, 2009: 36-43.
- [10] RUDRA A, DUBEY P K, JUTLA C S, et al. Efficient Rijndael encryption implementation with composite field arithmetic [C]// CHES 2001: Proceedings of the Third International Workshop on

Cryptographic Hardware and Embedded Systems. Berlin: Springer-Verlag, 2001: 171-184.

- [11] 张志峰,林正浩. AES 加密算法中 S-BOX 的算法与 VLSI 实现[J]. 计算机工程与应用, 2006, 42(19): 67-68.
- [12] 潘宏亮,高德远,张盛兵,等. 一种基于有限域求逆的 S-Box 实现算法[J]. 微电子学与计算机, 2006, 23(3): 109-111, 115.
- [13] 韩少男,李晓江. 实现 AES 算法中 S-BOX 和 INV-S-BOX 的高效方法[J]. 微电子学, 2010, 40(1): 103-107.
- [14] FISCHER V, DRUTAROVSKY M, CHODOWIEC P, et al. InvMixColumn decomposition and multilevel resource sharing in AES implementations [J]. IEEE Transactions on Very Large Scale Integration Systems, 2005, 13(8): 989-992.
- [15] JARVINEN K, TOMMISKA M, SKYTITA J. Comparative survey of high performance cryptographic algorithm implementations on FPGAs [J]. IEE Proceedings Information Security, 2005, 152(1): 3-12.