第36卷 第15期 计 算 机 工 程 2010年8月
Vol.36 No.15 Computer Engineering August 2010
·安全技术· 文章编号：1000—3428(2010)15—0134—03 文献标识码：A 中图分类号：TP309

基于协同特征的BIOS Rootkit检测技术

邓立丰[1]，王宏霞[2]
(1. 西南交通大学信息科学与技术学院，成都 630031；
2. 西南交通大学信息安全与国家计算网格实验室，成都 630031)

摘 要：对木马模型框架的理论及其协同隐藏模型进行研究和分析，给出BIOS Rootkit协同隐藏模型的具体形式化描述，提出一种在Windows环境下基于协同特征的BIOS Rootkit检测技术。针对现有BIOS Rootkit检测技术存在能查但不能正常恢复的问题，采用搜索特征码并动态恢复的技术解决。实验结果表明，该检测技术具有良好的可靠性、检测效率和完整性。

关键词：协同隐藏；BIOS Rootkit技术；检测

BIOS Rootkit Detection Technology Based on Cooperation Feature

DENG Li-feng[1], WANG Hong-xia[2]
(1. School of Information Science and Technology, Southwest Jiaotong University, Chengdu 630031;
2. State Laboratory for Information Security and Computing Grid, Southwest Jiaotong University, Chengdu 630031)

【Abstract】 To research and analyze the theory of framework for modeling trojans and trojans model about cooperative concealment, this paper gives BIOS Rootkit's formal model about cooperative and presents a BIOS Rootkit detection technology based on the character of cooperation in Windows environment. Current detection technology of BIOS Rootkit can detect but lacks the function of resuming, so the method uses characteristic of codes searching and dynamic resuming to solving it. Experimental results show that this is a detection method of higher reliability, efficiency and integrity.

【Key words】 cooperative concealment; BIOS Rootkit technology; detection

1 概述

作为恶意软件的一种特定类型，Rootkit近年来成为国内外计算机安全领域热门的研究课题，特别是新兴的Rootkit分支BIOS Rootkit，强调将传统的Rootkit技术和BIOS芯片相结合，其具有较强隐蔽性和较大破坏力，几乎可以躲过现有的任何计算机安全检测软件的检测，使计算机安全检测面临巨大的挑战。因此，对BIOS Rootkit检测技术的研究显得尤为重要。

2 BIOS Rootkit隐藏技术

Rootkit的概念起源于Unix/Linux操作系统，最初是指Unix/Linux系统中一组用于获取并维持Root权限的工具集。如今Rootkit是指能够持久或可靠地存在于计算机上的一组无法检测的程序和代码[1]。

BIOS是一组固化到计算机主板ROM芯片上的程序，它保存着计算机最重要的基本输入输出的程序、系统设置信息、开机上电自检程序和系统启动自举程序[2]。

BIOS的类型一般根据所在硬件位置来区分。如存在于主板中的BIOS一般称为System BIOS或Mainboard BIOS，它负责整个主板的运行；嵌入在显示适配器(显卡)中的BIOS称为Video BIOS；网络适配器(网卡)中的BIOS称为Netcard BIOS；SCSI控制卡和磁盘附加卡中也都存在相应的BIOS芯片[2]。

BIOS Rootkit是将Rootkit存放于BIOS芯片中然后在操作系统被加载运行之前第一时间获得系统控制权的一种技术。根据嵌入的模块位置不同可以分为：基于PCI模块的BIOS Rootkit，基于ISA模块的BIOS Rootkit，基于ACPI的BIOS Rootkit[3-4]等。

BIOS Rootkit比常规的Rootkit更难检测，其原因在于BIOS Rootkit相对于常规的Rootkit技术具有以下特点[4]：(1)能在第一时间获取系统主动权。(2)不在磁盘上留下痕迹。(3)能够重复感染已有操作系统或新装系统。(4)能对抗现有的几乎所有HIPS、杀毒、审计等安全软件。(5)难以清除。

3 基于协同特征隐藏的BIOS Rootkit模型

BIOS Rootkit协同隐藏是指通过Rootkit几个模块的相互协同作用实现特定代码和数据的隐藏。为了深入研究BIOS Rootkit，揭露其本质，本文以计算机程序及其输入输出为研究对象，参考木马模型框架[5]、木马协同隐藏模型[6]及文献[7]中的技术，然后给出BIOS Rootkit协同隐藏的形式化描述。

3.1 相关定义

为了方便讨论，下面给出文献[5]提出的一些相关定义。

定义1 计算机的状态、程序文本和图像等统称为表示，记为R，R是有限的。

定义2 E:R ($L|R$)，E是表示到环境的映射。$L|R$的含义

基金项目：四川省青年科技基金资助项目(07ZQ026-004)
作者简介：邓立丰(1983－)，男，硕士研究生，主研方向：Rootkitt检测技术，数字水印；王宏霞，教授、博士、博士生导师
收稿日期：2010-01-07 **E-mail**：Licplove@163.com

定义 3 $E(r)$的域：$names\ r=dom\ E(r)$，是可计算的和有限的，其中包含一些独立于 r 的名字。

定义 4 程序的含义是指程序在运行时完成了什么工作，用 $[P]$ 表示。如果 $r\in R$，r 中的程序 P，$[P]$ 的含义用$[\bullet]$: $R\to(R\to R)$ 表示。程序的运行将使一种表示变为另一种表示，在没有歧义的情况下，将 $\Omega=[E,R,[\bullet]$ 用 R 表示。

定义 5 程序 P 和 P'相等是指：$\forall r\in R,[P]r=[P']r$。

定义 6 考虑检测程序输出的过程所耗费的时间，定义 R 上小于线性时间可计算的关系为相似，用 $\sqcup$ 表示，相似关系是非传递的。

定义 7 对于大多数输入，如果 2 个程序 P 和 P' 能够产生相似结果，则认为它们是不可区分的，记作 $P\approx P'$，即：

$P\approx P'\text{iff}\ \ M\ \ r\in R:[P]r\sqcup[P']r$

为方便讨论，再引入一些符号：

l 是可以自然扩展到 $l_1,l_2,\cdots,l_n\in L^*$ 的有限序列；

$r\xrightarrow{l}r'$：$r\xrightarrow{l}r'$ iff $l\in names\ r$ and $r'=[E(r)l]r$。

3.2 BIOS Rootkit 协同隐藏形式化描述

设 P 表示一个 BIOS Rootkit，$P=(M,N,\overline{R1},\overline{R2})$。其中，$M$ 表示 P 在系统加载内核前的若干组成部分的集合，$M=\{M_i\mid i=1,2,\cdots,m\}$；$N$ 表示 P 在系统加载内核后的若干组成部分的集合，$N=\{N_i\mid i=1,2,\cdots,n\}$。

设 L_1^*为系统内核加载之前的所有代码序列，L_2^*为内核加载后的代码序列，然后 $r,\hat{r},r',\hat{r}'\in R$，可以得到 BIOS Rootkit 协同隐藏模型的形式化描述：

$$\wedge\ E(r)P\approx E(\hat{r})P\ ,\ L^*=L_1^*+L_2^*$$

$$\wedge\ r\sim\hat{r}\ ,\ M_i\in L_1^*,\ N_j\in L_2^*,\ \exists t$$

$$\wedge\ [E(r)P]r\xrightarrow{t}r'$$

$$\wedge\ [E(\hat{r})P]r\xrightarrow{t}\hat{r}'$$

$$\begin{cases}\wedge r'\sqcup\hat{r}'(t\in L^*)\\ \wedge r'\not\sqcup\hat{r}'(t\notin L^*)\end{cases}$$

通过上述模型可以看出，M 与 N 之间的协同作用使 P 实现了隐藏。

4 基于协同特征的BIOS Rootkit检测技术

从上一节介绍的模型形式化描述可知，BIOS Rootkit 是协同隐藏的一种类型。本文根据 BIOS Rootkit 各部分的协同关系，提出一种基于协同特征的 BIOS Rootkit 检测算法。

4.1 算法的设计思想

BIOS Rootkit 通过 Rootkit 几个模块的相互协同作用实现特定代码和数据隐藏。通过分析发现，BIOS Rootkit 一般至少分为 3 个模块：启动模块，隐藏模块，功能模块。启动模块负责获取系统控制权；隐藏模块负责对功能代码的保护，实现隐藏执行；功能模块负责实现某种具体且有价值的功能。它们之间的协同关系[7]如图 1 所示。

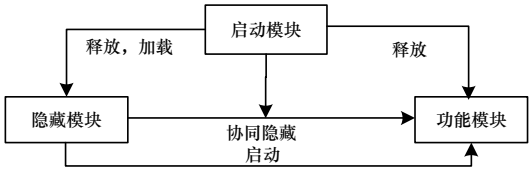

图 1 BIOS Rootkit 的协同隐藏关系

各个模块之间的协同关系，能得到很好保持的前提是启动模块一定要完整地实现其功能，如果启动模块在执行过程中遭到破坏，那么其他功能模块的执行也会受到很大的影响，甚至无法实现隐藏模块顺利加载运行及功能模块的释放。

本算法主要的思想是：通过详细分析 BIOS Rootkit 的启动模块，在第一时间改变启动模块的执行流程，从而让操作系统回到一种正常状态执行，这样便可以达到无法顺利加载并运行隐藏模块和功能模块的目的。

4.2 启动模块

启动模块很大程度上依附于整个计算机的引导过程，特别是在操作系统内核运行之前的过程，本算法是在 Windows 版本下的操作系统上实现的。基于 Windows XP 操作系统的计算机的引导过程可以简要归纳为：首先开电源，此时的 CS:IP 值为：ffff:0000，此处往往是条跳转指令，跳到 BIOS 中的初始部分代码去执行；然后执行 BIOS，BIOS 负责加电自检、硬件设备初始化之类的工作，这些工作完成后 BIOS 调用 INT 19h 中断服务去读主引导扇区的代码到内存的 0x0000:0x7C00 位置，这样 BIOS 的工作初步结束；接下来执行主引导扇区的代码，主引导扇区的代码功能包括：存放硬盘分区表，检查硬盘分区的正确性，确定活动分区号，读取相应操作系统的引导记录，检查操作系统引导记录的正确性、释放控制权给相应的操作系统。此时引导扇区的代码运行然后加载 NTLDR 文件，并把控制权给 NTLDR，这时操作系统开始启动，然后加载内核并且把控制权交给内核，之后便是内核完成操作系统的全部启动过程[8]。

通过对大部分 BIOS Rootkit 启动模块的分析，发现启动模块的代码会在 BIOS 调用 INT 19h 时去 HOOK INT 13h 中断服务(其中的 INT 19h 是 BIOS 引导自举的中断服务，负责将主引导扇区读到内存然后把控制权交给主引导扇区；INT 13h 是 BIOS 的磁盘操作服务中断，它可以为操作系统提供磁盘的读写等服务)。由于操作系统在读 NTLDR 文件到内存时用到了 INT 13h 中断服务，因此 BIOS Rootkit 的启动模块在操作系统运行时还能再次获得 CPU 的控制权，这样启动模块就可以通过搜索内存去劫持内核或 Detour BootDriver，然后加载并运行隐藏模块。虽然各种 BIOS Rootkit 可能在内核和 BootDriver 加载时劫持的地方不太一样，但是通过反汇编分析和动态调试发现，HOOK INT 13h 中断服务是它们一开始都要做的工作，通过这个 HOOK 便可以再次获得 CPU 的控制权。由于要实现原来的 INT 13h 的功能，因此在被 HOOK 后的 INT 13h 代码中(即启动模块的部分代码)存放了原来中断地址或中断地址的偏移量，这样就可以作为恢复 INT 13h 中断的根据。

从上面的分析可以发现，如果能让这个钩子还原，即让被 BIOS Rootkit HOOK 过后的 INT 13h 地址还原为原来的 INT 13h 地址，便可以让启动模块的功能失效，相应的隐藏模块和功能模块的功能也就失效了。这是本文检测算法所要解决的主要问题。

4.3 算法的具体设计

4.3.1 算法的问题及其解决方案

对启动模块的详细分析是算法设计的根本依据所在，本算法主要围绕 2 个关键问题进行分析并提出解决方案：

(1)检测 BIOS 的 INT 13h 是否被 HOOK。

(2)如果 INT 13h 被 HOOK 了，如何修复 INT 13h 的中断地址。

第(1)个问题的解决方法是：通过地址的比较可以看到BIOS 的 INT 13h 是否被 HOOK。如果被 HOOK，则再进行下一个步骤即还原 INT 13h 中断向量；如果没有被 HOOK，则说明不存在 BIOS Rootkit，不用进行下一个步骤的工作。

其中要设定一个阈值用于比较，然后确定 BIOS 的 INT 13h 是否被 HOOK。由于中断向量的地址存放在中断向量表(存放了中断的段地址和偏移地址)中，即 0000: 0000~0000: 0400 的地址中，中断服务的代码是属于 BIOS 的，因此这段代码是只读的代码，它会存放在内存的高地址中，即在 C000:0000～FFFF: 0000 的地址之间[9]，那么以段地址 C000 为阈值，然后用 INT 13h 中断向量表中的段地址与阈值比较，如果小于则说明存在 BIOS Rootkit，如果大于或者等于则说明不存在 BIOS Rootkit。

第(2)个问题的解决方法是：由于 HOOK 的 BIOS Rootkit 启动模块代码必然要用到原来的中断服务，因此启动模块代码中必定存在原来的中断地址，也必定存在远转移之类的指令，以便跳转到原来中断服务的程序中执行。通过搜索远转移之类的机器指令可以找到正确的 INT 13h 中断服务的地址，这样就可以修复 INT 13h 的中断向量表，也就相应地修复了被 HOOK 的操作系统。

一般指令机器码的组成形式是：操作码+操作数。要实现远转移，可以通过下面 3 种方法实现：jmp far，call far，中断服务请求(int n)[10]，而在 jmp far 和 call far 中又分直接和间接 2 种寻址方式，表 1 列出了这几种指令的具体形式及其机器码和对应操作码的解释。

表 1　5 种远(段间)转移的具体形式

名称	指令的形式	指令的机器码	操作码的解释
中断调用	int n	CD+中断号	操作码为中断号
段间直接跳转	jmp far 地址/标号	EA+偏移+段基址	操作码为段地址：段内偏移
段间间接跳转	jmp far dword ptr CS:[标号]	2E FF 2E+标号偏移	操作码为某个存储中断地址的变量在内存段中的偏移量
段间直接调用	call far 地址/标号	9A +偏移+段基址	操作码为段地址：段内偏移
段间间接调用	call far dword ptr CS:[标号]	2E FF 1E+标号偏移	操作码为某个存储中断地址的变量在内存段中的偏移量

从表 1 可以看出，只要找到这些指令的操作码并做相应的定位，再做正确的判断，就能很好地恢复 INT 13h 中断向量的地址。

4.3.2　算法流程

通过上面的分析可以很清楚地知道检测算法所依据的原理以及所要解决的问题，然后就可以画出详细的程序流程图，见图 2。

由于此算法是以协同隐藏的理论为依据，并结合了实际的 BIOS Rootkit 分析结果，因此通过这种检测方法可以很好地恢复 INT 13 中断向量的地址，保证了操作系统的完整性，同时也让 BIOS Rootkit 的隐藏模块和功能模块完全失效，从而达到检测和恢复的目的。

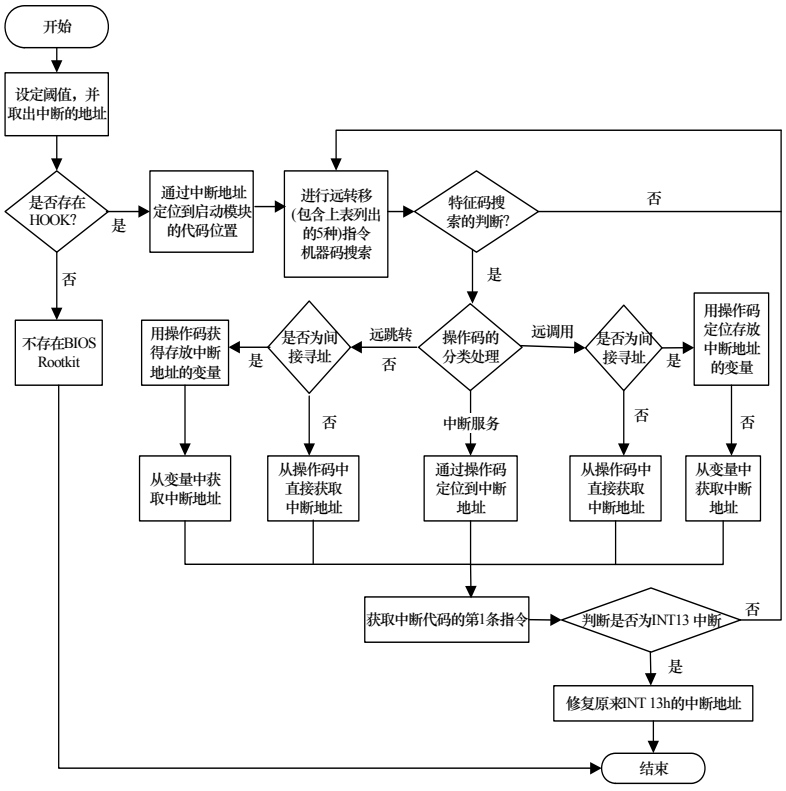

图 2　检测流程

5　实验结果

为了验证基于协同特征的 BIOS Rootkit 检测技术的性能，测试了 4 种不同的 BIOS Rootkit，包括 Icelord 的基于 ISA 模块的 BIOS Rootkit[11]、Cheng5103 的基于 PCI 模块的 BIOS Rootkit(Telnet 后门)[12]以及 2 个未公开的 BIOS Rootkit 程序 BRX 和 BRY，将实验所用恶意代码分别在装有 Windows XP SP2 系统的虚拟机上进行测试。检测结果与检测耗时汇总统计如表 2 所示。

表 2　实验结果

BIOS Rootkit 名称	使用原来中断服务所采用的转移方式	能否检测出 Rootkit 并进行正常化的恢复	耗时/s
Icelord 8.0	jmp far 地址/标号	Y	<2
Cheng5103	int n	Y	<2.5
BRX	call far dword ptr CS:[标号]	Y	1~2
BRY	jmp far dword ptr CS:[标号]	Y	1

从实验结果可以看出，这 4 种 BIOS Rootkit 都能被检测出，并且通过检测算法都能成功地使操作系统恢复正常，所以，基于协同特征的 BIOS Rootkit 检测技术具有很高的可靠性和很强的完整性。由耗时可以看出这种检测算法的检测效率很高。

6　结束语

BIOS Rootkit 检测技术目前处于起步阶段，这方面公开发表的论文很少，所以，成熟可靠的方法还有待于进一步研究。实验结果表明该检测算法具有可靠性、高效性和完整性。但该检测技术也存在一些不足，如在恢复中断服务方面存在一些误差。因此，要完全对抗 BIOS Rootkit 还需要将该技术与其他检测技术进行结合，使其更加有效。

(下转第 139 页)