第 37 卷 第 2 期
Vol.37 No.2

计 算 机 工 程
Computer Engineering

2011 年 1 月
January 2011

• 开发研究与设计技术 • 文章编号：1000—3428(2011)02—0251—02 文献标识码：A 中图分类号：TP311

基于行为特征的 BIOS Rootkit 检测

郭致昌，张 平，庞建民，郭浩然，崔 晨

(解放军信息工程大学信息工程学院，郑州 450002)

摘 要：针对 BIOS Rootkit 难以检测的问题，提出一种基于行为特征的 BIOS Rootkit 的检测方法。该方法通过研究 BIOS Rootkit 工作原理和实现技术，对 BIOS Rootkit 的行为特征进行归纳、定义和形式化描述，在反编译的过程中提取行为，根据提取的行为构成 BIOS Rootkit 的完整程度进行恶意性判定。实验结果证明，该方法能够有效检测主流的 BIOS Rootkit。

关键词：BIOS 安全；逆向工程；恶意代码

BIOS Rootkit Detection Based on Behavior Characteristics

GUO Zhi-chang, ZHANG Ping, PANG Jian-min, GUO Hao-ran, CUI Chen

(Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002, China)

【Abstract】BIOS Rootkit is difficult to detect. Aiming at the problem, this paper proposes a behavior characteristics-based BIOS Rootkits detection method. By studying the BIOS Rootkits' principles and key technologies, this method generalizes, defines and formally describes the behavior characteristics of BIOS Rootkit. Behaviors information is got in decompile phases, and the malicious level is judged according to the rate of a complete BIOS Rootkit formed by those behaviors. Experimental results prove that the method is effective to detect BIOS Rootkits.

【Key words】BIOS security; reverse engineering; malware

DOI: 10.3969/j.issn.1000-3428.2011.02.088

1 概述

当前，Rootkit 与 Anti-Rootkit 的较量已经深入到计算机系统的核心及底层器件，计算机 BIOS 芯片是两者较量的新战场[1-2]。在 BIOS 中植入 Rootkit 会优先获得权限，能够绕过常规检测，具有隐蔽性好、查杀难度大、抗清除等特点。因此，研究人员提出了完整性检查与特征匹配相结合[3]和动态监测[4]的检测方法。前者能精确检测出 BIOS 文件的微小改动，但是维护不同厂商不同型号 BIOS 文件的检验和是一项繁琐的工作，而且简单的代码混淆手段会使特征码失效。后者是在特定位置和特定时刻对关键系统调用进行监控，但是实现难度较大，检测的时机和范围非常有限，难以做到通用。BIOS Rootkit 的巨大危害和检测手段缺乏的现状使计算机系统的安全状况堪忧，亟待新的检测方法。本文结合软件逆向工程技术[5]与恶意代码检测的思想，提出并实现一种检测 BIOS Rootkit 的方法。

2 BIOS Rootkit 工作原理和实现技术

一般来说，一个 BIOS Rootkit 可以划分成 3 个部分：

(1)权限获取模块。主要功能是在 BIOS 中完成代码嵌入，实现对 BIOS 的劫持。一般有 3 种方案：1)拦截和修改 BootBlock。2)将 BIOS Rootkit 作为 BIOS 的整体或部分功能模块嵌入到 BIOS 中。3)设置硬件中断，同时修改 int 01h 中断处理程序的入口地址为 Rootkit 地址，当断点触发时 Rootkit 就被触发。

(2)权限保持和隐藏模块。在 BIOS 与引导程序、操作系统内核之间传递控制权时，该模块实现 BIOS Rootkit 接管权限传递实现系统正常启动，以及穿透操作系统内核实现自身隐藏。该模块主要通过对读入内存的操作系统内核打补丁或者对关键的函数做 inline hook 的形式实现。

(3)payload 模块。每个 Rootkit 都带有一个或多个 payload，BIOS Rootkit 也不例外。该模块实现方式比较多，可以调用系统的 API 函数直接实现，也可以通过加载驱动程序或者释放文件的方式实现。

3 BIOS Rootkit 行为定义

3.1 BIOS Rootkit 行为选择

根据对上述 BIOS Rootkit 3 个模块的功能定义及实现技术的分析，提取其独有操作作为它们的行为特征模式。本文将 BIOS Rootkit 的行为特征模式归纳为 4 类：

(1)中断向量的重定位行为。自然语言描述为：首先将原来的中断处理程序的入口地址保存到某个位置，然后修改中断向量的入口地址使其指向 Rootkit 代码所在的内存地址，最后在特定的时机将原中断处理程序的入口地址写回。

(2)修改系统内核行为。常用的方法有 2 种：1)对操作系统内核进行补丁操作。自然语言描述为：从某个位置开始搜索特定的指令(字符串)，找到之后对其修改，通常将其修改成一条无条件跳转指令(E9 xx xx xx xx)。如果没有搜索到，则不做任何操作，返回到正常的流程继续执行。2)对操作系统内核进行 inline hook。自然语言描述为：从某个位置开始搜索特定的目标指令(字符串)，如果搜索到，则先将目标指令另存到某个位置做恢复用，之后再修改该指令。通常将其修改成一条无条件跳转指令，在特定的时机将修改过的指令恢复。如果没有搜索到，则不做任何操作，返回到正常流程继续执行。

(3)实现 payload 功能行为。虽然实现方式众多，但是其共同的特点是实现 payload 需要依赖操作系统的某种特殊结

基金项目：国家“863”计划基金资助项目(2009AA01Z434)；河南省重大科技攻关计划基金资助项目(092101210501)

作者简介：郭致昌(1985－)，男，硕士研究生，主研方向：逆向工程；张 平，副教授；庞建民，教授；郭浩然，博士研究生；崔 晨，硕士研究生

收稿日期：2010-06-29 **E-mail：**zwgg2010@gmail.com

构。例如，调用 API 函数行为描述为：利用特定的内核变量或符号计算得到一个内存地址，通过该内存地址取得 dll 的加载的基地址，遍历 dll 搜索到需要的 API 函数地址，通过地址进行调用。

(4)加密解密行为。自然语言描述为：从某个位置开始取出一个值；对取出的值配合密钥做运算，运算结果存储到另一个位置；判断加/解密是否完成，完成则加/解密行为结束，未完成则对取值和存储结果的位置进行调整。

特殊情况：

(1)在 BIOS 中一般的设备只包含 16 位实模式下执行的程序，不会用到 32 位保护模式下执行的程序，但是显卡 PCI 例外。

(2)在 BIOS 中极少对中断向量进行重定向，但显卡 BIOS 为了显示需要对 int 10 进行修改。

3.2 BIOS Rootkit 行为的形式化描述

BIOS Rootkit 的行为用一个五元组来定义。

<行为名称，行为长度，行为构造，行为区位图，扩展域>

行为名称域为以“ fun- ”开始的字符串，用于标识行为。

行为长度域为非负的整形常数，表示行为构造域的元素个数。

根据定义的粒度不同，行为又分成父行为和子行为 2 类。行为构造域的规则范式如下：

父行为={子行为名称 1，子行为名称 2，…，子行为名称 *n*};

子行为={语句 1，语句 2，…，语句 *m*};

语句=Call exp|Jmp exp|Cmp exp,exp|Assign;

Assign=m[exp]:=exp|r[exp]:=exp|tmp:=exp;

exp=exp op exp|m[const]|r[const]|const;

op=+|-|*|/|~|<<|>>||;

const=0|1|2…;

上述范式中，m[]表示内存单元，r[]表示寄存器。

行为区位图域由 0、1 和 x 的组合的序列定义，序列中的 0 或 1 与行为构造的元素(子行为名称或语句)一一对应，构成行为构造域的位图。x 不与任何元素对应，起分隔符的作用，将行为构造域划分成不同的区。位图域中的第 *i* 位为 1 表示行为构造中的第 *i* 个元素位置固定，为 0 时只有当第 *i*-1 位或者 *i*+1 位也为 0 时才有实际意义。位图域在同一个区中的 *i*, *i*+1, …, *i*+*k* 位都为 0，那么行为构造中对应 *i*，*i*+1, …, *i*+*k* 位置的元素互换位置后仍然属于同一个行为。

扩展域对行为构造中的元素作限定。

例如，中断向量的重定向行为定义为：

<fun-intHookA,4,00x00,P,{n=(1:0x2e),IVT:=0}>

P={m[tmp1]:=m[n*4+IVT]

m[tmp2]:= m[n*4+2+IVT]

m[n*4 + IVT] := m[tmp1]

m[n*4+2+ IVT]:= m[tmp2]}

其中，n 表示中断向量号，在 BIOS 初始化阶段，在内存地址为 0 的位置建立的是临时中断向量表。检测中只关注受影响的中断向量，为了对中断向量重定向行为做出完备的描述，还需要添加行为模板，此处不再列举。

4 基于行为特征的 BIOS Rookit 检测

4.1 检测系统框架

基于行为特征的 BIOS Rootkit 检测系统框架如图 1 所示，各模块功能为：BIOS 采样及分解模块针对不同厂商的 BIOS 对 BIOS 镜像文件采样并分解为一些具有一定格式的功能区段。逆向分析模块负责识别 BIOS 镜像文件中不同功能区段的程序入口点，然后进行相应的解码，解码后功能区段中的代码变换成语义等价的低级中间表示，并建立局部控制流图。语义分析模块利用反编译技术对中间表示进行语义提升，变换为高级的中间表示。行为识别模块对高级中间表示进行行为提取，如果在某个范围内的高级中间表示与行为库中的某个行为模式之间能建立映射关系，那么认为提取到该恶意行为。综合判定模块根据提取到的行为特征做进一步的优化，尽可能排除导致“ 误报 ”的因素，最后给出合理判定结果。

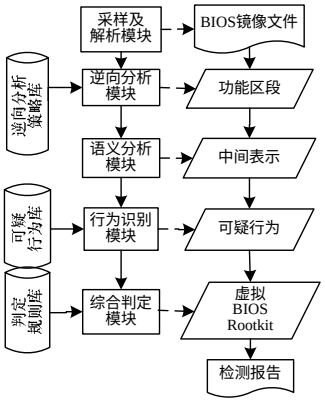

图 1 检测系统框架

4.2 解码流程控制

BIOS 镜像文件在嵌入 Rootkit 后结构变得更复杂，对 BIOS 模块的逆向分析不同于一般的可执行程序的逆向分析。BIOS 中含有多种格式的功能区段，在库中存储不同模块文件的入口信息和模块的结构信息，如 PCI 模块含有头结构，也可能是没有头结构的纯代码片段。通常 BIOS 中只含有实模式下执行的 16 位代码，由于 Rootkit 的存在，模块中可能含有 32 位保护模式下执行代码，在解码时用 16 位和 32 位 2 种模式对代码试解码，再根据试解码结果的正确性进行选择。由于 Rootkit 植入的干扰，BIOS 进行逆向解码不可能一次完成。此处采用多次线性扫描和按控制流方式解码相结合的方式对模块进行解码。如果遇到跳转指令并且目标地址在整个模块地址范围之外，则认为当前过程结束。在对解码做调整时，如果遇到没有与其他指令有任何关系的单条孤立指令，则将其删除。解码流程如图 2 所示。

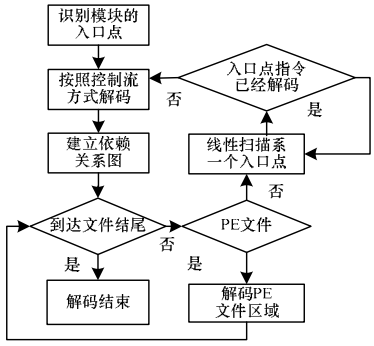

图 2 解码流程

4.3 BIOS Rootkit 判定

行为识别模块提取的行为是一些孤立行为，而且某些可疑行为在特定型号的 BIOS 功能模块中属于正常行为，所以，仅根据某个孤立的行为不足以断言 BIOS Rootkit 的存在。在

(下转第 255 页)