

基于硬件资源访问控制的固件恶意行为研究^{*}

张翠艳, 张 平, 胡 刚, 薛 亮

(解放军信息工程大学 信息工程学院, 郑州 450002)

摘 要: 固件与传统的应用软件一样, 都有可能存在木马、后门、逻辑炸弹等具有恶意行为的代码。由于固件具有硬件相关性、任务执行的阶段性与高内聚性等特点, 使得传统的程序恶意行为描述方法不能适用于固件程序。探讨了固件程序及固件恶意行为的特点和本质特征, 描述了一种基于硬件资源访问控制策略的固件程序恶意行为形式化建模和检测方法, 并对该方法的有效性进行了实验验证。

关键词: 固件恶意行为; 用户意愿; 硬件资源访问控制; 恶意固件检测

中图分类号: TP309

文献标志码: A

文章编号: 1001-3695(2011)07-2709-03

doi:10.3969/j.issn.1001-3695.2011.07.088

Research on malicious behavior of firmware based on hardware resources access control

ZHANG Cui-yan, ZHANG Ping, HU Gang, XUE Liang

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

Abstract: As same as the traditional application and system software, firmware also faced the risk of malicious code like hobbyhorse, back door, logical bomb and so on. Firmware exhibited strong cohesion and hardware relativity, which make the malicious action in firmware to be different from that in the traditional software. This paper analyzed the specificities of firmware and the malicious behavior about it, then expatiated the essence of the malicious behavior of the firmware, and presented a firmware formal definition and detecting method which was based on the hardware resources access control policy. Experimental results prove that the method is effective to detect the malicious firmware.

Key words: firmware malicious behavior; user's intention; hardware resources access control; malicious firmware detection

0 引言

固件是存储在永久存储器件中的二进制程序。随着集成电路制造技术的发展, 固件可以存储的芯片范围比较广泛, 如 ROM、PROM、Flash、Soc、SoPC 芯片以及具有内部存储器的微控制器等。固件是与硬件紧密联系的软件, 通常工作在特权模式, 在以微处理器为核心的电子设备中, 为上层软件有效地使用硬件设备提供调用接口, 是电子系统中的重要组成部分。计算机系统中, BIOS 和扩展 ROM 中的程序、以 ARM 为核心的嵌入式应用系统中的启动代码 boot loader、单片应用系统中的内嵌控制程序以及 PCI 控制器扩展 ROM 中的代码等都是典型的固件。

固件作为一类特殊的软件, 与传统应用软件和系统软件一样, 都有可能存在木马、后门、逻辑炸弹等具有恶意行为的代码, 如 ACPI Rootkit^[1]、PCI Rootkit^[2] 将恶意代码植入 Flash 芯片实施恶意攻击, King 等人^[3] 利用固件程序来实现特定恶意行为的影子模式攻击。

固件是系统的底层软件, 拥有更多的操作权限, 可以留下或植入更多可能的安全隐患, 如引入后门、篡改文件、修改硬件配置等。传统应用层上行之有效的安全工具不能检测和清除固件中的恶意代码, 固件恶意代码的出现动摇了现有的安全防

护机制。Borg^[4] 指出: 恶意固件能够一直潜伏在电子设备中, 一旦满足其执行的条件时则可能导致信息系统瘫痪, 甚至能够在攻击者的操纵下控制电子设备。

本文首先对传统的恶意行为定义进行探讨, 指出传统恶意行为定义在描述固件恶意行为方面的不足, 结合成熟的用户意愿的思想, 对固件恶意行为进行形式化定义, 并给出了一种基于硬件资源访问控制的固件恶意行为检测方法。最后对公开的一个固件恶意程序 IceLoad 进行实验分析, 验证了该检测方法的有效性。

1 恶意代码的传统定义

Cohen^[5] 1984 年将计算机软件病毒定义为“一种程序, 它可以感染其他程序, 感染的方式为在被感染程序中加入计算机病毒的一个副本, 这个副本可能是在原病毒基础上演变过来的”, 并得出了复制性病毒不可判定的结论。Cohen 对病毒的定义与文献[6]中关于包含病毒、蠕虫、木马的定义及其形式化描述一样, 都是将恶意程序外在特征和采用技术作为恶意行为判定的标准。然而, 技术是没有恶意与善意之分, 外部特征与病毒相似的代码也可能是一段正常的程序, 如同样是利用 Raw Socket 向网络发送程序, 黑客可能会利用它来完成信息的窃取, 而普通用户则可能利用它来完成正常的业务数据传输。

收稿日期: 2010-12-03; 修回日期: 2011-01-07 基金项目: 国家“863”计划资助项目(2009AA01Z434)

作者简介: 张翠艳(1983-), 女, 江苏泰州人, 硕士研究生, 主要研究方向为恶意代码分析(zcy728@163.com); 张平(1969-), 女, 吉林通化人, 副教授, 博士, 主要研究方向为程序安全性分析; 胡刚(1984-), 男, 新疆吐鲁番人, 硕士研究生, 主要研究方向为固件逆向工程; 薛亮(1983-), 男, 山东蓬莱人, 助理工程师。

因此,恶意程序具有不确定性质。

性质 1 不确定性质。通过分析代码的外部特征或执行功能的方法不能够确定一个程序是否具有恶意行为。

利用恶意程序的代码特征和所采用的特殊技术来定义恶意行为是不严谨的,并不能反映恶意行为的本质特征。为了克服这种定义的不足,文献[7]给出了基于程序规范的恶意代码定义,即判断一个程序是否有恶意行为的依据是程序的行为是否在程序规范允许的范围内。但是这样的定义有两个严重的不足:a)扩大了恶意行为的范围,程序中有可能存在超出程序规范但对系统或用户没有恶意影响的功能,这样的功能既不能造成任何性能上的影响也不会带来任何安全上的问题,不能称为恶意行为。b)不能包含大部分的恶意行为,合法功能的特殊组合可以产生恶意的操作,如数据库访问非一致性漏洞就属于此类情况。还有一些定义将一些典型的行为作为判断程序恶意性的标准,如冒充用户身份、通过共享区域来穿越保护区域边界、篡改文件、改变统计特性等。因此,很难从程序功能本身来判断恶意行为的存在,即恶意行为具有不可判定性质。

性质 2 不可判定性质。不存在一个独立算法能够准确地检测恶意行为的存在。

有关不可判定性质的详细说明参见文献[8]。由于检测恶意行为存在的独立算法不存在,研究人员开始从恶意行为本身之外来寻求恶意行为的准确定义,引入了用户意愿的思想。一段代码之所以具有恶意行为是因为在某一前置条件下执行这段代码将会给用户带来不愿意得到的结果,即违背了用户的意愿或期望。

定义 1 恶意程序。通常所说的恶意程序是指具有显式作用(文档中描述的或期望的已知效应)又有隐式作用(文档中没有描述的或非期望的效应)的程序。

恶意行为是指恶意程序中实现隐式作用的逻辑。用户意愿指用户对程序的期望的效应。何鸿君等人^[9]提出的一种广义病毒的形式定义充分体现了用户意愿的思想,将用户意愿描述为用户对系统文件的授权访问,并证明了该定义涵盖大部分具有隐式作用的病毒。

2 固件恶意行为描述

2.1 固件代码的特征

与传统软件中的恶意代码相比,固件恶意代码有三个不同之处:a)固件程序存储于只读存储器中,所以 Cohen^[10]关于病毒的形式化定义已经不能用于固件恶意代码。由于存储载体具有只读的特点,固件中的恶意代码不能够随心所欲地复制传播。虽然有的固件存储于可读写的 Flash 芯片中,但是在单个芯片内的复制不如文件感染那样有意义。b)固件与传统的软件相比,抽象级最低,与硬件的距离最近,其功能实现通过直接控制硬件来完成。因此固件程序具有硬件相关性,用户意愿不能通过对文件的访问授权来体现,这种硬件相关性向传统恶意代码的定义提出了新的挑战。c)固件代码存储于各类计算机系统中,这些系统随着应用的不同而具有不同的硬件配置,其恶意代码的表现形式也是五花八门。一般情况下,系统 A 中的恶意代码通常难以在系统 B 中体现其恶意性,而系统 C 中的正常功能在系统 D 中或许就是一个危险系数极高的恶意行为。固件恶意代码表现形式的这种不确定性和多样性增加了恶意代码分析与检测的工作量,同时也为固件恶意行为的准确

定义增添了困难。第 3 章将从固件的特殊性出发,引入用户意愿的思想,尝试对固件恶意行为进行新的定义。

固件(firm-ware)通常由中断(interruption)、中断向量表(interruption_table)和主程序(main_process)组成。主程序本质上也是中断(一般为复位中断处理过程)。每一个中断通常包含一个或多个功能模块(modules),功能模块完成与某一特定硬件交互的功能,每一个功能模块可以同时包含在不同的中断处理过程中。一般的固件结构如图 1 所示,可以描述为

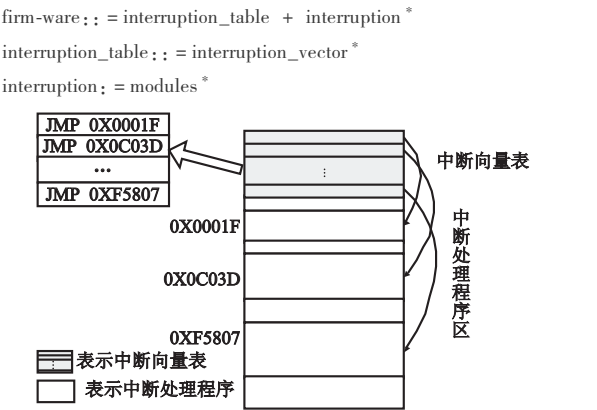


图 1 固件结构图

2.2 固件恶意行为的形式化描述

固件具有硬件相关性,固件的正常功能和恶意功能都是通过直接使用硬件资源来实现。硬件资源在计算机系统中表现为 IO 空间和存储空间。代码对硬件资源的使用是通过读写 IO 空间和存储空间来完成的。文中为了描述的方便,假定 IO 空间和存储空间统一编址,并统称为存储空间。

利用有限状态机可以准确地描述一个固件程序的行为,将计算机建模为一个有限状态机 $M = (Q, I, \delta, q_0)$, Q 表示存储空间的状态, I 表示处理器的指令集, δ 是状态迁移函数,表示指令的语义, $q_0 \in Q$ 表示处理器重置时的存储空间的初始值。一段固件程序 π 在计算机 M 上运行的结果可以表示为 $q' \leftarrow \delta^*(q, \pi)$ 。在对固件进行恶意性分析时,通常不需要对固件行为进行如此精细的刻画,只需关注固件行为中影响系统安全策略的部分,即读、写、执行存储空间的固件行为。

定义 2 固件行为。固件行为 action 是固件程序中访问硬件资源的操作, $action \subseteq interruption \times module \times mode \times mem$, $action = (int, module, m, mem) \in action$, $int \in interruption$, $module \in module$, $m \in mode$, $mem \in mem$,表示中断 int 过程的模块 module 以模式 m 访问了存储空间 mem。

定义 3 用户意愿。用户意愿是用户发布的固件对存储空间的访问授权说明,即用户允许固件中断的模块(module)以模式(mode $\in \{read, write, execute\}$)访问存储空间(mem),可以表示为四元组 $W = (interruption, module, mode, mem)$ 。

用户定义的所有意愿组成的集合称为用户意愿集 $I = \{w \mid w = (interruption, module, mode, mem)\}$ 。

定义 4 固件恶意行为。固件恶意行为是固件行为中超出用户意愿的部分,固件行为 $action = (int, module, m, mem)$ 是恶意的当且仅当 $(int, module, m, mem) \in action \wedge (int, module, m, mem) \notin I$ 。

3 固件恶意行为检测方法

固件恶意行为检测方法可以分为三个部分:模块划分、访

问授权和检测算法。

固件恶意行为检测首先要分析固件的模块结构与功能特征,将固件的中断处理例程划分为不同的功能模块,每一个模块通常以高度内聚的方式完成一个单一的功能,如 ACPI 电源管理模块、PCI 设备初始化模块等。功能的单一决定了所使用硬件资源的相对集中性。模块划分是将中断处理例程的逻辑空间 R 划分成 m 个不相重叠的区域,显然:

- a) $module_i \cap module_j = \Phi, i \neq j, i, j \in N$, 每个阶段不存在重叠的区域。
- b) $\bigcup_{i=1}^m module_i$, 中断处理例程中的每一行代码都属于一个功能模块。

访问授权就是定义用户意愿的过程,决定固件程序不同功能模块可以访问的硬件资源。根据定义 3,一次授权可以用一个四维向量表示,即 (interruption, module, mode, mem)。其中, (mem, mode) 为访问权限向量,为访问权限空间 R^2 中的一个点, (interruption, module) 代表固件程序中一个特定功能模块,访问授权就是在权限空间和模块集里建一个多对多的映射: $mode \times mem \rightarrow interruption \times module$ 。

所有授权的集合构成用户意愿集 $I \subseteq mode \times mem \times interruption \times module$ 。

恶意行为的检测就是在给定的用户意愿集下,监测固件代码的行为是否违反了用户意愿。将待检测固件程序置入一个可控的执行环境,如 Bochs、QEMU 等。加载并运行固件程序,实时监控程序的指令执行情况,记录当前程序所处的模块,跟踪硬件访问指令的行为,并判断行为是否属于用户意愿,若不属于,则该行为是恶意行为。检测算法如下:

算法名称: fwmDetect

input: 固件 P , 用户意愿集 I

output: 恶意行为集: $m\text{-action} = \{w \mid w = (int, module, m, mem) \in action \wedge (int, module, m, mem) \notin I\}$

1. $m\text{-action} \leftarrow \Phi$, $module \leftarrow$ 初始模块

2. while 指令被执行

3. if 指令是模块切换指令 then

4. $module \leftarrow$ 下一个模块 $module$

5. else if 固件行为 w 发生 then

6. if $w \notin I$ then

7. $m\text{-action} \leftarrow m\text{-action} \cup w$

8. end if

9. end while

算法 fwmDetect 描述了基于硬件资源访问控制策略的固件恶意行为检测方法的基本原理,在具体的应用中可以与程序分析中的静态和动态分析方法相结合,文中使用了动态分析中的多路分析方法^[11]。

4 实验分析

IceLord 在 2007 年公开了一款 BIOS Rootkit 实现的相关技术,并给出了其实现的二进制演示文件 IceLord.exe。IceLord.exe 在 NT 上运行,可以在 ISA 模块中植入恶意程序,恶意程序的逻辑保存在 IceLord 的资源文件 leaving.bin 中,其详细实现可参考文献[12]。

实验中采用 IceLord 实现的 BIOS Rootkit 作为实验样本,

使用资源编辑工具 eXeScope 提取 leaving.bin 文件,并通过编译器 Xeltek-280U 将其植入 PCI 网卡 RTL8139 的扩展 ROM 中,这样就得到了一个固件恶意代码的原型,该原型在 BIOS 自检时会在磁盘中创建一个能够在 OS 加载后自动运行的 protector.sys 文件。

固件恶意行为检测程序 test.exe 是在固件逆向分析平台 amPro 的基础实现的,amPro 是自主开发的二进制分析平台,平台支持 ARM、x86、MCS-51 等多种指令系统,并实现了一个基于虚拟指令语言的指令模拟器 amPro_emulator.dll,可以实现二进制代码的动态分析。test.exe 加载网卡扩展 ROM 中的程序并动态执行,检测跟踪指令的执行情况,实时监测固件的行为是否属于用户意愿集。整个检测框架如图 2 所示。

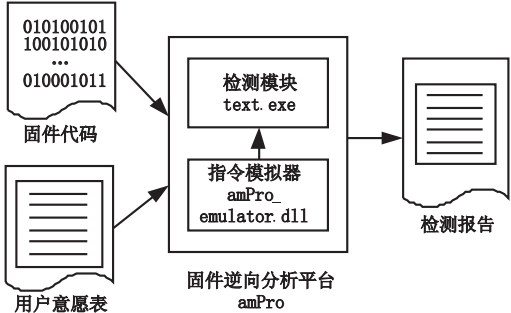


图2 检测框架图

由于 RTL8139 扩展 ROM 的主要功能是 RTL8139 控制器的初始化,因此扩展 ROM 能够访问的空间是分配给设备的 IO 空间和存储空间,通过读取端口 0xCF8 和 0xCFC 可以获得空间范围,并生成用户意愿配置文件,在实验机上运行 test-gi userauth.ini 的输出结果如图 3 所示。

C:\>test -gi userauth.ini

user's intention action set I test program based-on ampro inst_emulator			
id	access type	authorized mode	memory range
1	IO	<R, W>	<[B300-B3FF]>
2	MEM	<R, W>	<[FDEFFC00,FDEFFCFF]>
3	MEM	<R, W, E>	<D8000,D8040>

图3 用户意愿图

可以编辑 userauth.ini 文件,添加新的用户意愿,由于实验程序是网卡扩展 ROM 中的一个简单固件,所以使用自动生成的用户意愿集即可。运行 test-i userauth.ini-f leaving.bin-oreport.txt,得到检测结果如图 4 所示:

C:\>test -i userauth.ini -f leaving.bin -o report.txt

malicious action set I test program based-on ampro inst_emulator			
address	access type	mode	memory range
089	MEM	<R>	<[4C-4D], [4D-4F]>
093	MEM	<W>	<[4C-4D]>
09A	MEM	<W>	<[4D-4F]>
0AD	MEM	<W>	<[34-35], [35-36]>
0B2	MEM	<R>	<[34-35]>
0BC	MEM	<W>	<[35-36]>
0C3	MEM	<W>	<[4E-4F]>
0CA	MEM	<E>	<[FE6F2]>
176	MEM	<W>	<[34-35], [35-36]>
00D	MEM	<E>	<[FE3FE]>
0ED	MEM	<E>	<[FE3EF]>

图4 分析结果

检测程序发现该恶意固件修改了 13H 和 19H 中断向量地址,分析 leaving.bin 的源码可知,该固件通过修改 13H 和 19H 中断向量向操作系统植入 protector.sys。实验结果显示,该检测方法检测出了在网卡 RTL8139 规定之外的访问硬件资源的操作。

高,原因是文献[5]中的算法只充分利用了纹理区域的视觉不敏感性,而本文算法额外考虑到了边界和黑暗区域的视觉不敏感性,因此能嵌入更多的秘密信息。从理论上讲,纹理越强的图像,本文算法的嵌入容量越大。文献[9]指出,一个好的隐写算法在保持大嵌入容量的同时,PSNR 值应该在 40 dB 以上。由于充分利用了 HVS 的特性,本文算法的 PSNR 效果也较好。在表 1 中可发现,除了 baboon 等纹理特性非常强的图像外,其他图像的平均 PSNR 值在 41 以上,符合文献[9]中对于优秀方案的评定。Baboon 图像较低的 PSNR 值是由于其纹理特性极强,嵌入了大量的信息,也正是由于这种强纹理,确保了在视觉特性上隐写图像不会有明显变化。

在计算复杂性上,本文算法与文献[5]算法都要进行小波变换,不同的是主要集中在对小块分类中的除法运算。文献[5]方案中由于计算对比度,对每个小块进行 48 次除法;而本文方案只对每个小块进行 2 次求平均值的除法运算,因此本文算法比文献[5]算法更适合实际应用。

表 1 本文方案与文献[5]方案实验对比

载体图像 512 × 512	文献[5]方案		本文方案	
	容量/bit	PSNR/dB	容量/bit	PSNR/dB
Lena	590 848	40.608 4	658 666	42.5
baboon	668 416	39.256 1	840 547	38.2
airplane	572 160	41.589 4	641 845	42.1
man	598 016	40.952 5	763 813	40.4
house	531 200	43.221 7	711 397	41.6
couple	638 720	38.006 1	662 131	42.5
peppers	612 864	40.044 5	675 361	42.3

4 结 束 语

本文在研究有关图像复杂度描述方法的基础上,提出了一种利用小波平均值描述图像复杂度特性的方法,并结合模函数的方法设计了新的隐写算法,实验结果显示本方案取得了较好的性能。本文虽然提出了新的描述图像复杂度的方法,但本文

(上接第 2711 页)

5 结 束 语

由于固件程序的硬件相关性,代码执行的阶段性和高内聚性使得传统的程序恶意行为定义不能涵盖固件恶意行为,本文在分析固件程序及固件恶意行为特点的基础上,讨论了固件恶意行为的本质特征,并在传统恶意行为定义的基础上引入用户意愿的概念,提出了一种新的关于固件程序恶意行为的形式定义和检测方法。与现有的关于恶意行为的定义一样,由于恶意行为的不确定性和不可判定性的限制,文中的定义不能包含所有形式的固件恶意行为,但是该定义在传统恶意行为的定义基础上,充分考虑了固件的特殊性,涵盖了大部分固件恶意行为。实验结果表明,本文提出的方法确实可行。

参考文献:

[1] HEASMAN J. Implementing and detecting an ACPI BIOS Rootkit: Blackhat DC[R]. 2006.
[2] HEASMAN J. Implementing and detecting a PCI Rootkit: Blackhat DC [R]. 2007.
[3] KING S T, TUCEK J, COZZIE A, et al. Designing and implementing malicious hardware [C]//Proc of the 1st Usenix Workshop on

方案只是单纯从视觉特性这方面考虑,如何更精确地设计出系统地描述图像复杂度的方法仍然是今后研究的重点和难点。

参考文献:

[1] WU D C, TSAI W H. A steganographic method for images by pixel-value differencing[J]. *Sample Recognition Letters*,2003,24(9): 1613-1626.
[2] KAWAGUCHI E, EASON R O. Principle and application of BPCS-steganography[C]//Proc of SPIE: Multimedia Systems and Applications. 1998:464-472.
[3] ZHANG X, WANG S. Steganography using multiple-base notational system and human vision sensitivity [J]. *IEEE Signal Processing Letters*,2005,12(1): 67-70.
[4] KANG Z W, LIU J, HE Y. Steganography based on self-organizing competitive NNS and HVS[C]//Proc of the 5th International Conference on Distributed Computing and Applications for Business, Engineering and Sciences. Shanghai: Shanghai University Press, 2006: 395-397.
[5] 刘劲,康志伟,何怡刚. 一种基于小波对比度和 LSB 的密写[J]. *电子学报*,2007,35(7): 1391-1393.
[6] REN Jun-ling, MA Zhi-feng. Information hiding algorithm for palette images based on HVS[C]//Proc of IEEE International Conference on Wireless Communications, Networking and Information Security. 2010:354-358.
[7] 黄达人,刘九芬,黄继武. 小波变换域图像水印嵌入对策和算法[J]. *软件学报*,2002,13(7): 1290-1297.
[8] THIEN C C, LIN J C. A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function [J]. *Pattern Recognition*,2003,36(12): 2875-2881.
[9] CHEDDAD A, CONDELL J, CURRAN K, et al. Digital image steganography: survey and analyses of current methods[J]. *Signal Processing*,2010,90(3): 727-752.
[10] TOET A, RUYVEN L, VELATON J. Merging thermal and visual images by a contrast pyramid[J]. *Opt Engineering*,1989,28(7): 789-792.

Large-Scale Exploits and Emergent Threats. 2008.
[4] BORG S. Securing the supply chain for electronic equipment: a strategy and framework[R]. 2009.
[5] COHEN F. Computer viruses-theory and experiments[J]. *Computers and Security*, 1987,6(1): 22-35.
[6] CHESS D M, WHITE S R. An undetectable computer virus[C]//Proc of Virus Bulletin Conference. Orlando:[s. n.], 2000.
[7] NECULA G C. Proof-carrying code[C]//Proc of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. Paris France: [s. n.], 1997:106-119.
[8] BISHOP M. Computer security art and science[M]. Singapore:Pearson Education, 2005:435-436.
[9] 何鸿君,罗莉,董黎明,等. 广义病毒的形式定义及识别算法[J]. *计算机学报*,2010,33(3): 562-568.
[10] COHEN F. Computational aspects of computer viruses[J]. *Computers and Security*, 1989,8(4): 325-344.
[11] 陈凯,冯登国,苏璞睿. 基于延后策略的动态多路径分析方法[J]. *计算机学报*,2010,33(3): 493-503.
[12] IceLord. BIOS Rootkit: welcome home, my lord! [EB/OL]. (2007-05-11). <http://blog.csdn.net/icelord/archive/2007/05/11/1604884.aspx>.