ISSN 1001-9146
CN33-1339/TN

JOURNAL OF HANGZHOU DIANZI UNIVERSITY

杭州电子科技大学学报

ISSN 1001-9146

2011 5
Vol. 31 No.5 Oct. 2011

杭 州 电 子 科 技 大 学 学 报
2011 年 10 月　　（总 133）　　第 31 卷第 5 期

目　　次

期刊基本参数:CN33 – 1339/TN* 1981* b* A4* 188* zh* P* ¥6.00* 500* 46* 2011 – 10

JOURNAL OF HANGZHOU DIANZI UNIVERSITY

Vol. 31, No. 5 Sum. 133 Oct. 2011

CONTENTS

第31卷第5期
2011年10月

杭州电子科技大学学报
Journal of Hangzhou Dianzi University

Vol. 31, No. 5
Oct. 2011

一种Linux平台下的可执行文件防篡改方法

姜 斌[1],张 君[2]

(1. 杭州电子科技大学通信工程学院,浙江 杭州 310018;2. 浙江省电子信息产品检验所,浙江 杭州 310007)

摘要:保证可执行文件的完整性对于整个系统的安全性是非常重要的。该文提出了一种可执行文件防篡改方法,该方法的基本思想是:通过安全钩子捕获可执行文件的执行请求;在捕获执行请求之后,计算该文件当前的哈希值,并与预期哈希相比较。如果两者一致,则说明可执行文件是可信的,允许运行;否则是不可信的,拒绝运行。以Linux平台为基础实现了原型系统,实验验证,该方案能够有效保证可执行文件的完整性。

关键词:完整性;安全模块;信息安全

中图分类号:TP309 文献标识码:A 文章编号:1001-9146(2011)05-0087-04

0 引 言

随着信息技术的飞速发展,信息安全事件已经深入到人们生活的方方面面[1]。调查表明,在各类信息安全事件当中,恶意代码是最重要的安全威胁之一[2]。恶意代码要传播,必须要进行几个特定的操作,以最典型的病毒和木马为例,其在传播的过程中,必然要改写其它可执行文件,以达到感染的目的。因此,如果能有效检测到病毒木马对其它可执行文件的篡改,则可以遏止病毒木马的传播过程。基于这种思想,本文提出了一种可执行文件防篡改方法。这种方法的基本思想是:在任何一个可执行文件运行之前,先检查其是否被篡改。如果被篡改了,则说明该文件不可信,拒绝其执行;如果没有被篡改,则允许其执行。由此达到防止被篡改的(恶意)软件运行的目的。Linux平台下的实验结果表明达到了预期效果。

1 Linux平台下可执行文件防篡改架构

要实现可执行文件防篡改架构,有两个关键步骤:第一步,在任何一个可执行文件运行之前,先捕获其运行请求;第二步,在捕获到运行请求之后,判定该文件是否被篡改。要实现第一步,可以采用钩子机制。在Windows平台下采用相应的Hook函数即可,Linux平台下,则可以通过Linux安全模块(Linux Security Module,LSM)实现;要实现第二步,只要计算可执行文件的哈希值即可。如果可执行文件的当前哈希值与该文件的预期哈希值是一致的,则说明文件没有被篡改,是可信的;否则说明被篡改,是不可信的。

1.1 Linux内核安全模块

LSM是一个轻量级通用的访问控制框架。它通过在内核当中的特点位置增加检测点,并在检测点插入钩子函数。这样,当用户需要施加额外的访问控制的时候,可以改写这些钩子函数,从而获得安全增强[3]。一个访问文件节点inode的访问过程如图1所示。在经过了错误检查、访问控制DAC检查之后,就在Linux内核试图对内核对象进行访问之前,LSM钩子调用一个LSM模块所提供的函数。该函数根据策略引擎进行判断,要么允许此次访问,要么拒绝访问并强制返回一个错误。在LSM当中,对如

收稿日期:2011-07-20
基金项目:浙江省教育厅基金资助项目(Y200803422)
作者简介:姜斌,(1980-),男,浙江衢州人,助理研究员,网络优化.

下一些内核对象插入了安全钩子:任务钩子、程序钩子、文件系统钩子、管道/文件/socket 相关钩子;文件操作钩子;网络数据包相关钩子;网络设备相关钩子;信号量/共享内存/消息队列钩子;单个消息钩子等。可见,LSM 的功能还是比较强大的。

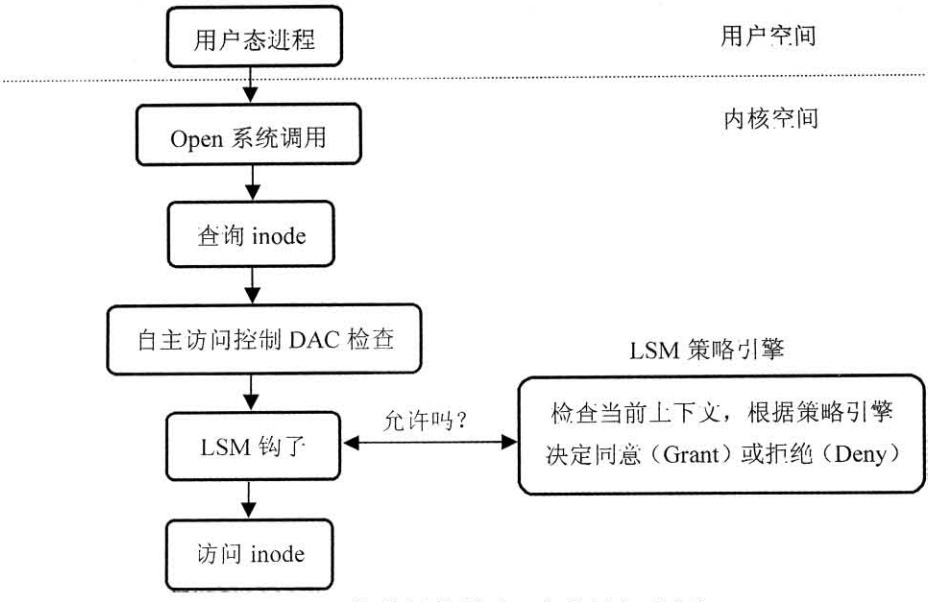

图 1 LSM 的挂钩结构(以文件访问为例)

1.2 系统架构

在 LSM 的基础上,Linux 平台下可执行文件防篡改架构如图 2 所示。

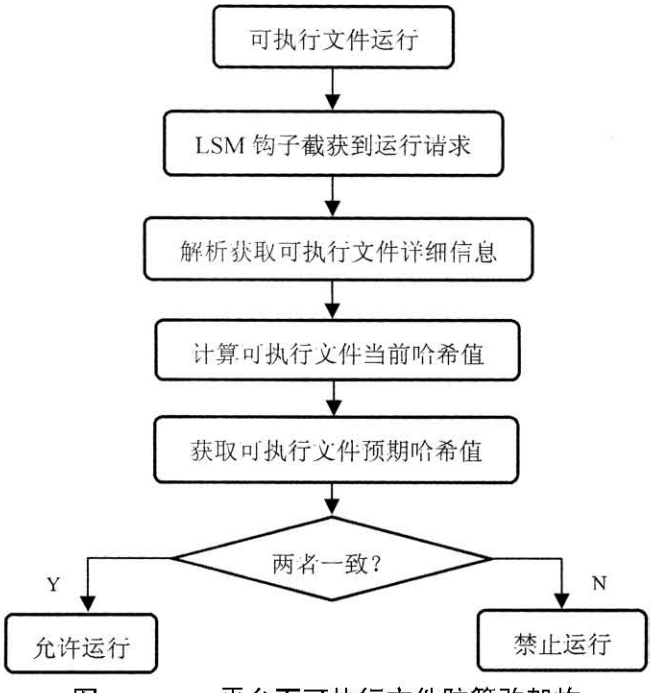

图 2 Linux 平台下可执行文件防篡改架构

整个架构的工作流程是这样的:(1)首先,某个可执行文件被用户运行;(2)LSM 钩子截获到该文件的运行请求;(3)相应的 LSM 钩子函数被执行。该函数将解析请求运行文件的详细信息——包括绝对

路径文件名、参数等等;(4)根据步骤(3)当中获取的可执行文件绝对路径文件名,计算该文件的当前哈希值;(5)获取该文件的预期哈希值(通常保存在外部的 U 盘当中);(6)将文件的当前哈希值与预期哈希值相比较,如果两者一致,则允许该可执行文件的运行,否则拒绝其运行。

2 系统实现

2.1 LSM 钩子设置

在图 2 的架构当中,为了截获可执行文件的运行请求,需要进行 LSM 钩子设置,这是通过设置 bprm_check_security 函数钩子实现的。bprm_check_security 钩子函数原型是:

int (* bprm_check_security) (struct linux_binprm * bprm)。其中,参数 bprm 包含了基本的 linux_binprm 结构体信息,例如:要执行的应用程序的真实名称、命令行参数信息等等。通过在 LSM 内核模块文件当中添加如下代码,可以设置 bprm_check_security 钩子:

```
static struct security_operations jetway_security_ops = {
. bprm_check_security = appprotec_check_security, };
```

这样,任何时候当某个应用程序开始运行之后,在 Linux 内核调用实际的加载器加载其运行之前,会自动转到定义的 appprotec_check_security 函数。appprotec_check_security 首先根据参数"struct linux_binprm * bprm"获取文件的相关信息,特别是其绝对路径文件名、参数信息等,其中 bprm -> interp 就给出了绝对路径文件名;然后,内核根据获取的绝对路径文件名打开相应的文件,计算其哈希(MD5);最后,内核从 U 盘读取该文件的预期哈希:如果两者不一致,则返回 -1 拒绝文件的运行;否则返回 0 允许其运行。

2.2 USB 驱动

在图 2 的架构当中,进行哈希值比较的时候,需要从 U 盘读取文件的预期哈希值。这就需要编写特定的 USB 驱动。为什么不直接使用 Linux 自带的 USB 驱动呢? 这是因为:需要对 U 盘进行判定,只有是特定的 U 盘,才相信其存储的预期哈希值,否则如果读取任意 U 盘的话,那攻击者完全可以伪造假的预期哈希值从而使得架构失去作用。从 U 盘读取预期哈希值的过程是:首先从内核判定插入的 U 盘是不是允许的 U 盘(这可以通过 id 号,以及特殊扇区的特殊信息来保证),如果是允许的 U 盘,则读取 U 盘内部名为 expectedhash 的文件,并解析该文件获取预期哈希值。

USB 本身有 4 种传输方式:等时传输、中断传输、控制传输和猝发传输。USB 使用 1ms 的时间帧来传输数据。在每次传输之前,HC 产生一个帧头 Start Of Frame。每次传输的时候,如果有等时数据存在,则等时数据最先被传输。HCD 需要确保:有足够的时间用来完成所有的等时传输和中断传输数据,然后使用剩下的时间来完成控制传输和猝发传输数据[4]。根据上述 USB 驱动特点,就可以对 USB 驱动框架进行填写,开发实际的 USB 驱动了。描述 USB 驱动程序的结构体是系统定义的标准结构 struct usb_driver,在 struct usb_driver 中,name 用来告诉内核模块的名字;probe 和 disconnect 是函数指针,当设备与在 id_table 中的变量信息匹配时,函数被调用;id_table 用来告诉内核模块支持的设备。更详细的 USB 开发可以参见文献 4、5。

3 实验和讨论

3.1 实验结果

Linux Redhat 内核 2.4 上实现了一个原型系统,构造了一个名为 appproect.ko 的 LSM 驱动,在其中实现了 int (* bprm_check_security) (struct linux_binprm * bprm) 钩子。在 root 权限下,使用"insmod appprotect.ko"加载的内核,然后插入具有预期哈希信息的 U 盘。启动/bin/ls,可以发现,正常列出了文件和目录信息;之后,使用编辑器更改/bin/ls 当中任意位置的任意个字节,再次运行,系统显示"operation not permitted"。实验达到了预期结果。

经过多次实验统计,系统引起约 1% 的额外负载,这主要是计算哈希值所引起的。不过,1% 的负载对系统性能影响不大,可以接受,用户几乎感受不到时间的延迟。

3.2 讨论

首先需要指出的是，虽然本文在 Linux 平台实现的原型系统，但该架构仍然可以应用到 Windows 平台上面。当应用到 Windows 平台时，其基本流程类似于 Linux 平台（如图 2 所示），唯一不同的是，Linux 平台采用 LSM 捕获可执行文件运行请求，而在 Windows 平台下，可以通过设置 Windows 钩子的方法来达到相同目的。要设置 Windows 钩子，可以调用 SetWindowsHookEx() 函数。其次需要指出的是，本文提出的方法是在可执行文件运行之前，先判定该文件是否已经被篡改。也就是说，本文的方法并不能“阻止篡改”，而只能“发现篡改”，并在“发现篡改”之后，调用备份文件进行主动恢复，从而达到保护的目的。之所以不采用“阻止篡改”的方法，是由于采用“阻止篡改”的方法，就需要判定当执行写入操作时，执行写入的进程究竟是一个正常进程，还是一个病毒或者木马进程，而这种判定从理论上是不可行的：因为不存在一个算法，能够判定任意一个文件是否是病毒或者木马。基于这种原因，才采用了“发现篡改”的方法。注意到在架构当中，任何一个可执行文件在刚刚发出执行请求，而并没有创建进程实体之前，其完整性就被检查了，因而这种方案不仅能够发现对文件的篡改，而且能够有效避免系统感染病毒或者木马，保证了系统的安全性。

4 结束语

本文提出了一种可执行文件防篡改框架，该框架在任何一个可执行文件实际运行之前，先通过计算哈希值判定其是否被非法篡改。如果被篡改，则禁止其运行；否则允许其运行。本文以 Linux 平台为基础，实现了一个原型系统，达到了预期效果。下一步，该架构可以进一步扩展，即：在可执行文件进行完整性哈希比较时，针对指定的配置文件进行完整性比较。

参考文献

[1] 国际计算机网络应急技术处理协调中心. CNCERT 互联网安全威胁报告[EB/OL]. http://www.cert.org.cn/UserFiles/File/201101monthly.pdf, 2011-02-01.

[2] 国际计算机网络应急技术处理协调中心. CNCERT 互联网安全威胁报告[EB/OL]. http://www.cert.org.cn/UserFiles/File/201102monthly.pdf, 2011-03-01.

[3] Wright C, Cowan C, Smalley S, et al. Linux Security Modules: General Security Support for the Linux Kernel [C]. San Francisco: Proceedings of the 11th USENIX Security Symposium, 2002: 1-14.

[4] Intel. Universal Host Controller Interface (UHCI) Design Guide [EB/OL]. http://download.intel.com/technology/usb/UHCI11D.pdf, 2011-05-31.

[5] Corbet J, Rubini A, Kroah-Hartman G. Linux Device Drivers (Third Edition) [M]. O'Reilly Media Press: Cambridge Massachusetts, 2005: 10-14, 50-64.

An Approach for Protecting Linux Platform-runable from Being Tampered

JIANG Bin[1], ZHANG Jun[2]

(1. *School of Communication Engineering, Hangzhou Dianzi University, Hangzhou Zhejiang* 310018, *China*;
2. *Zhejiang Provincial Testing Institute of Electronic Information Products, Hangzhou Zhejiang* 310007, *China*)

Abstract: Protecting the integrity of applications is important to the security of an information system. This paper introduces a framework for keeping the applications from being tampered. The basic idea of the framework is as the following: firstly, capturing the running request of an application according to security hooks; then, calculating the current hash the application and compare it with the expected hash. If they match, it means the application is trusted and the running request is granted; otherwise untrusted and the running request is denied. We implement a prototype in the Linux platform, and experimental results show that our approach can efficiently protect the integrity of applications.

Key words: integrity; security module; information security

杭州电子科技大学学报

2011年10月　第31卷　第5期　双月刊

JOURNAL OF HANGZHOU DIANZI UNIVERSITY

Vol. 31 No. 5 Oct., 2011 Bimonthly

Started in 1981

HANGZHOU DIANZI KEJI DAXUE XUEBAO

Sum No.133

主办单位:杭州电子科技大学

编辑出版:杭州电子科技大学学报编辑部
浙江省杭州市下沙高教园区
电话:0571-86915100　电传:0571-86919103
邮政编码:310018
E-mail:hdxb@hdu.edu.cn

国内发行:杭州电子科技大学学报编辑部
天津市大寺泉集北里别墅17号
邮政编码:300385

印　　刷:杭州地质印刷有限公司

Sponsor:Hangzhou Dianzi University

Editor and Publisher:Editorial Department of Journal of HDU

Address:Shiasha Higher Education Zone
Hangzhou, Zhejiang, 310018,
China

Distributed:Editorial Department of Journal of HDU

Printed:Printing House of Huaxing
(Hangzhou, China)

国内统一刊号: ISSN1001-9146 / CN33-1339/TN　(公开发行)

定价:6.00元