基于 LKM 机制的 Linux 安全模块的研究

李晓丽
(南通大学,江苏 南通 226000)

摘 要:近年来,Linux 系统由于其出色的稳定性、灵活性和可扩展性,以及较低廉的成本,受到计算机工业界的广泛关注和应用。但在安全性方面,Linux 内核只提供了自主访问控制以及部分安全机制,这对于 Linux 系统的安全性是不够的,影响了 Linux 系统的进一步发展和更广泛的应用。文中在深入研究 LKM 和 HOOK 技术的基础上,针对目前 Linux 系统安全审计方面的不足,设计了一个 Linux 安全日志模块。当入侵者通过某用户账户进入 Linux 服务器系统并尝试修改文件时,系统会自动生成包含用户信息的日志记录文件。该模块适用于监管长期稳定运行、配置较少需要改动的 Linux 服务器系统。经系统实测,该安全日志模块能及时有效地记录恶意用户对系统文件的访问或篡改,为系统的安全审计工作提供有用信息。

关键词:可装载内核模块;钩子函数;安全模块;系统调用;短信报警
中图分类号:TP309 **文献标识码**:A **文章编号**:1673-629X(2016)06-0097-04
doi:10.3969/j.issn.1673-629X.2016.06.021

Research on Linux Security Module Based on LKM Mechanism

LI Xiao-li
(Nantong University, Nantong 226000, China)

Abstract: In recent years, the Linux system has been widely concerned and applied in the computer industry because of its excellent stability, flexibility and scalability, and low cost. But in terms of security, the Linux kernel only provides access control as well as some security mechanisms. This is not enough for the security of the Linux system, which affects the further development of the Linux system and its wider application. In view of the current problems on security auditing of Linux system, a Linux security log module is designed in this paper on the basis of the research of LKM and HOOK technology. When an intruder enters the Linux server system through a user account and tries to modify the file, the system will generate a log file containing the user's information automatically. The module is suitable for monitoring the long-term stable operation of the Linux server system, whose configuration is less need to change. By the actual measurement, the security log module can record the user's access or tampering with the system file in time and effectively, and provide useful information for the security auditing of the system.

Key words: LKM; hook function; security module; system call; SMS alarm

0 引 言

Linux 是一款免费的操作系统,用户可以通过网络或其他途径免费获得,并可以任意修改其源代码。因为它符合 IEEE POSIX.1 标准[1],移植性好,目前已被广泛使用,很多大中型企业的应用服务都是构筑在其之上,例如 Web 服务、数据库服务、集群服务等等。但是,相比于 Windows 操作系统,它的安全问题一直没有得到很好的解决。基于此,文中在深入研究可加载内核模块(Loadable Kernel Modules, LKM)的基础上,使用 HOOK 技术,构建了一个文件访问日志记录模块[2-3]。如果入侵者通过某用户账户进入服务器系统并尝试修改文件,则会在相应文件目录下生成一个含有相关信息的日志记录,这样系统就可以在第一时间定位该用户。

1 LKM

Linux 是单内核的操作系统[4],即整个系统内核都运行于一个单独的保护域中。相比于微内核的操作系统,单内核由于把所有的系统功能模块都集中到一起,系统的性能和速度都非常好,但是可扩展性和维护性

收稿日期:2015-08-08 **修回日期**:2015-11-12 **网络出版时间**:2016-05-05
基金项目:国家自然科学基金资助项目(61373169);南通大学自然科学基金资助项目(12Z057,13Z040)
作者简介:李晓丽(1977-),女,硕士研究生,高级实验师,CCF 会员,研究方向为信息安全。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20160505.0828.068.html

就相对较差。为了弥补单内核的这个缺点,Linux 操作系统使用 LKM 机制[5-6],可以在运行时动态地更改 Linux。

可动态更改是指允许内核在运行时动态地向其中插入或从中删除代码。这些代码包括相关的子例程、数据、函数入口和函数出口,被一并组合在一个单独的二进制镜像中,即所谓的可装载内核模块中,或被简称为模块。这是一种区别于一般应用程序的系统级程序,它主要用于扩展 Linux 的内核功能。LKM 的优点是基本内核镜像可以尽可能小,可以最小化内核的内存占用,只加载需要的元素,可选的功能和驱动程序可以利用模块形式再提供[7]。模块允许用户方便地删除和重新载入内核代码,也方便了调试工作,无须重新编译内核。而且当热插拔新设备时,可通过命令载入新的驱动程序。

2 相关技术

2.1 系统调用

操作系统中的状态分为内核态和用户态。操作系统的主要功能是为管理硬件资源和为应用程序开发人员提供良好的环境来使应用程序具有更好的兼容性。为了达到这个目的,内核提供一系列具备预定功能的多内核函数,通过一组称为系统调用(system call)的接口呈现给用户。一般用户程序只在用户态下运行,有时需要访问系统核心功能,这时通过系统调用接口进行系统调用[8-9]。系统调用把应用程序的请求传给内核,调用相应的内核函数完成所需的处理,将处理结果返回给应用程序[10]。

换言之,系统调用即为用户空间访问内核的一种方式,其具体执行过程如图 1 所示[11]。

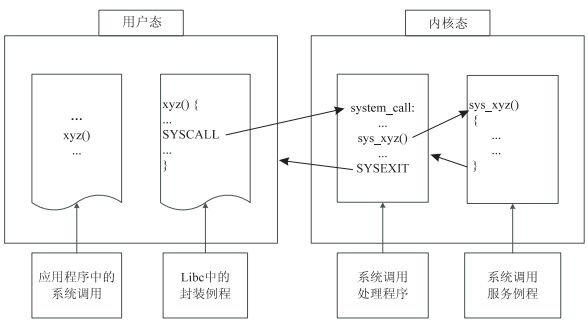

图 1 系统调用过程示意图

(1)在应用程序中调用用户空间的库函数;

(2)库函数在执行一系列预处理后,取得系统调用号,传递相应的参数并执行软中断指令 INT 产生中断;

(3)Linux 系统进行地址空间的转换和堆栈的切换,执行 SAVE_ALL 宏定义,从而保存任务现场;

(4)根据系统调用号,从系统调用表找到对应系统调用处理程序的入口地址;

(5)执行系统调用对应的处理程序;

(6)执行 RESTORE_ALL 宏定义,恢复系统调用前的任务现场并返回用户模式。

2.2 HOOK

HOOK(钩子)是一种特殊的消息处理机制。钩子可以监视系统或进程中的各种事件消息,截获发往目标窗口的消息并进行处理。这样,用户就可以在系统中安装自定义的钩子,监视系统中特定事件的发生,完成特定的功能,比如截获键盘、鼠标的输入,屏幕取词,日志监视,等等。

HOOKING(挂钩)实际上是一个处理消息的程序段,通过系统调用,把它挂入系统。每当特定的消息发出,在没有到达目的窗口前,HOOK 函数就先捕获该消息,亦即 HOOK 函数先得到控制权。这时 HOOK 函数既可以加工处理(改变)该消息,也可以不作处理而继续传递该消息,还可以强制结束消息的传递。

内核 HOOKING 原理如图 2 所示。

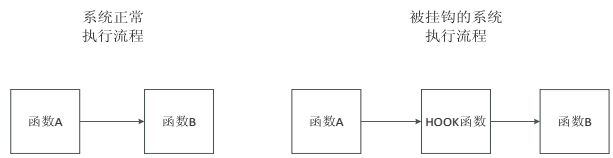

图 2 内核 HOOKING 原理示意图

从图 2 可以看出,HOOK 可用来扩展(或削弱)一个子程序的功能,也可用来修改系统应用程序编程接口(Application Programming Interface, API)的运行效果。

2.3 Linux 系统调用 HOOK 相关技术

Linux 系统调用所使用的 HOOK 技术主要有如下几种:

1)Kernel Inline Hook。

目前流行和成熟的 Kernel Inline Hook 技术就是修改内核函数的 opcode,通过写入 jmp 或 push ret 等指令跳转到新的内核函数中,从而达到修改或过滤的功能。具体过程参见图 3。

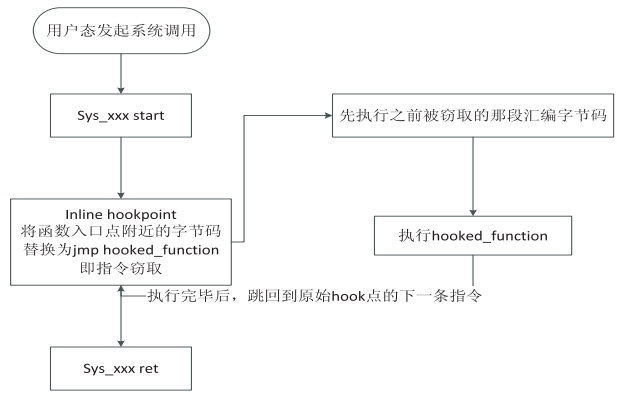

图 3 Kernel Inline Hook 流程图

2)利用 0x80 中断劫持。

要对系统调用(sys_call_table)进行替换,必须要获取该地址后才可以进行替换。但是 Linux 2.6 版的内核出于安全的考虑,没有将系统调用列表基地址的符号 sys_call_table 导出。考虑到系统调用都是通过 0x80 中断来进行的[12-13],故通过查找 0x80 中断的处理程序来获得 sys_call_table 的地址。其基本步骤如下:

(1)获取中断描述符表(Interrupt Descriptor Table,IDT)的地址;

(2)从中查找 0x80 中断(系统调用中断)的服务例程(8 * 0x80 偏移);

(3)搜索该例程的内存空间;

(4)从中获取 sys_call_table(保存所有系统调用例程的入口地址)的地址。

3)利用 kprobe 机制。

kprobe 是一个动态地收集调试和性能信息的工具。它从 Dprobe 项目派生而来,几乎可以跟踪任何函数或被执行的指令以及一些异步事件。

4)LSM Security 钩子技术。

Linux 安全模块(Linux Security Module,LSM)是 Linux 内核的一个轻量级通用访问控制框架。它使得各种不同的安全访问控制模型能够以 Linux 可加载内核模块的形式实现出来。用户可以根据其需求选择适合的安全模块加载到 Linux 内核中,从而大大提高了 Linux 安全访问控制机制的灵活性和易用性。

目前已有很多著名的增强访问控制系统移植到 LSM 上实现,包括:POSIX. 1e capabilities,安全增强 Linux(SELinux),域和类型增强(the Domain and Type Enhancement,DTE),Linux 入侵检测系统(Linux Intrusion Detection System,LIDS)。

3　安全模块设计

在 Linux 系统中,一般存在多个用户。为了保障系统安全,要监视系统中的所有非法活动并将其记录到系统日志中。基于 Linux 下“一切皆文件”的思想[14],任何入侵者进入系统后的所有动作都是针对文件的操作。因此,当入侵者通过某个用户账户进入系统并尝试修改文件时(文中暂不考虑新增文件和删除文件的情形),为了能够在第一时间定位该用户,在深入研究 0x80 中断机制的基础上,通过在 Linux 系统内核调用中增加文件访问/修改记录日志模块的方法来实现上述功能[15]。

模块分三个阶段进行设计,其工作流程图如图 4 所示。

(1)安装阶段。

在模块安装阶段,首先对环境变量进行初始化,然后查找系统调用列表 sys_call_table[ ]的基地址并记录系统调用服务例程(如 sys_open,sys_close,sys_read,sys_write 等)的入口地址,接着将 sys_open 函数的入口指针替换成函数指针(my_sys_open)。

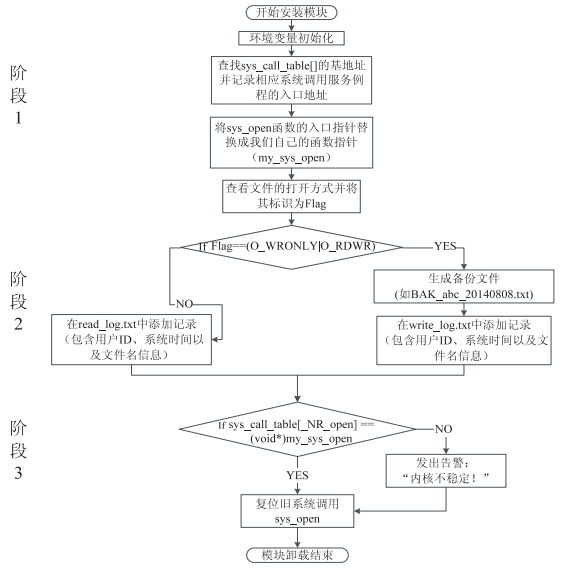

图 4　模块工作流程图

(2)操作阶段。

初始化完成后,这时会执行函数 my_sys_open。通常,文件被打开的方式一般有以下三种模式:O_RDONLY/O_WRONLY/O_RDWR。根据这三种打开方式,my_sys_open 函数会决定将生成的文件访问日志记录添加到 read_log. txt 中还是 write_log. txt 中。

当文件以 O_WRONLY 或 O_RDWR 方式打开时,系统首先会对该文件进行拷贝以生成备份文件(名称格式为 BAK_文件名_系统时间,用于后期文件的恢复),然后会在 write_log. txt 日志文件中添加一条包含有用户 ID、当前系统时间以及原始文件名信息的记录。例如,原始文件名为“abc. txt”,当前系统时间为 2014 年 8 月 8 日 23 点 56 分 22 秒,用户的 ID 为 001200,则生成的备份文件名为 BAK_abc_20140808235622. txt,并在 writelog. txt 日志文件中生成以下记录:001200#20140808235622# abc. txt。

当文件以 O_RDONLY 方式打开时,则会在 read_log. txt 文件中添加如下记录:001200# 20140808235622 # abc. txt。

(3)模块卸载阶段。

在移除文件访问日志模块时,需要将 my_sys_open 的地址和 sys_open 现在的地址进行对比。如果相同,则复位旧系统调用(使用 sys_open 的原始地址来替代现在的 sys_open 地址);如果两个地址不同,则表明系统调用列表的完整性已遭到破坏,内核处于不稳定状态,此时会给用户发出相关信息告警。

4 关键函数

(1)模块的初始化函数 init_module()。

```
// 查找 sys_call_table[ ]的基地址
find_sys_call_table( );
//记录系统调用服务例程的原始地址
original_open = (void * ) sys_call_table[ _NR_open];
original_write = (void * ) sys_call_table[ _NR_write];
original_close = (void * ) sys_call_table[ _NR_close];
original_read = (void * ) sys_call_table[ _NR_read];
//用系统调用函数 my_sys_open 替代 sys_open
sys_call_table[ _NR_open] = (void * ) my_sys_open;
```

(2)监控函数 my_sys_open()。

```
// 如果文件以只读或可读写方式打开
if( flags = = (O_WRONLY | O_RDWR))
{
backup_file(filename, flags, mode); //生成备份文件
print_id(USER_ID);
strcpy(timelog, USER_ID);
print_time(USER_TIME);
strcat(timelog, USER_TIME);
strcat(timelog, filelog);
//将用户 ID、系统时间、文件名记录到 write_log. txt 中
Write_file( "write_log. txt", timelog);
}
//如果文件以只读方式打开
else
{
print_id(USER_ID);
strcpy(timelog, USER_ID);
print_time(USER_TIME);
strcat(timelog, USER_TIME);
strcat(timelog, filelog);
// 将用户 ID、系统时间、文件名记录到 read_log. txt 中
write_file("read_log. txt", timelog);
}
returnoriginal_open(filename, flags, mode);
```

(3)模块的卸载函数 cleanup_module()。

```
if(sys_call_table[ _NR_open] ! = (void * ) my_sys_open)
{
printk(KERN_ALERT "The system is in an unstable state! \
n");
}
else
{
sys_call_table[ _NR_open] = (void * ) original_open;
}
```

5 结束语

Linux 是服务器操作系统中最常用的,因为其拥有高性能、高扩展性、高安全性,受到了越来越多人的追捧;但是针对 Linux 服务器操作系统的安全事件也非常多。文中在深入研究 LKM 和 HOOK 技术的基础上,设计了一个能够在非法用户访问或修改文件时自动生成日志记录的信息模块。该模块适用于监管长期稳定运行、配置较少需要改动的 Linux 服务器系统。经系统实测,该安全日志模块能及时有效地记录恶意用户对系统文件的访问或篡改,为系统的安全审计工作提供有用信息。

接下来的工作是在上述研究基础上,再设计一个短信报警模块。当 write_log. txt 日志文件发生改变时,可以通过移动网关第一时间发送报警信息到管理人员的移动终端上。另外还需继续改进模块,增加对新增文件和删除文件的监控功能,并进一步优化设计,提高系统性能。

参考文献:

[1] Bovet D P, Cesati M. Understanding the Linux kernel[M]. [s. l.]: O'Reilly Media, Inc. , 2005.

[2] 吴 娴,钱培德. 基于 LSM 框架构建 Linux 安全模块[J]. 计算机工程与设计,2008,29(24):6281-6284.

[3] 张 浩,刘乃琦. Linux 安全模块框架的研究和安全日志的实现[J]. 计算机应用研究,2006,23(6):135-137.

[4] Hildebrand D. An architectural overview of QNX[C]//Proc of USENIX workshop on microkernels and other kernel architectures. [s. l.]: USENIX, 1992:113-126.

[5] Degoyeneche J M, Desousa E A F. Loadable kernel modules [J]. IEEE Software, 1998, 15(1):65-71.

[6] 徐 敏,熊盛武. Linux 2.6 内核下 LKM 安全性研究[J]. 电子设计工程,2011,19(12):21-24.

[7] Corbet J, Rubini A, Kroah-Hartman G. Linux device drivers [M]. [s. l.]: O'Reilly Media, Inc. , 2005.

[8] 罗 宇,邹 鹏,邓胜兰. 操作系统[M]. 北京:电子工业出版社,2011.

[9] 李云雪,苏智睿,王晓斌. 基于 Linux 安全模块的通用框架研究与实现[J]. 计算机工程,2005,31(3):105-107.

[10] 张丽芬,刘美华. 操作系统原理教程[M]. 北京:电子工业出版社,2013.

[11] 吴国伟,李 莹,姚 琳. Linux 内核分析与高级教程[M]. 北京:清华大学出版社,2012.

[12] Rajagopalan M, Hiltunen M, Jim T, et al. System call monitoring using authenticated system calls[J]. IEEE Transactions on Dependable and Secure Computing, 2006, 3(3):216-229.

[13] Matthew N, Stones R. Beginning Linux programming[M]. [s. l.]: John Wiley & Sons, 2011.

[14] Parker S. Shell scripting: expert recipes for Linux, Bash and more[M]. [s. l.]: John Wiley & Sons, 2011.

[15] Pratik A. Linux kernel module for security enhancement[M]. [s. l.]: ProQuest, 2007.