基于 LKM 的 Linux 安全检测器的设计与实现*

袁　源, 罗　红, 戴冠中, 吕　鹏
(西北工业大学 自动化学院, 陕西 西安 710072)

摘　要: 分析两个 Linux 后门工具的实现机制, 指出它们的原理及其危害; 针对 Linux 操作系统的特点提出两种保护 Linux 内核的方法, 即单模块内核方式和带安全检测的 LKM 方式。给出了基于 LKM 的 Linux 安全检测器的实现方法, 实验表明, 该安全检测器能有效地记录 LKM 后门工具以及病毒对系统的攻击和非法访问, 能够帮助系统管理员维护 Linux 操作系统的安全。

关键词: Linux; 可装载内核模块; 后门工具; 系统调用
中图法分类号: TP393.08　　**文献标识码:** A　　**文章编号:** 1001-3695(2005)07-0131-03

Design and Implementation of Linux's Security Monitor Based on LKM

YUAN Yuan, LUO Hong, DAI Guan-zhong, LV Peng
(College of Automation, Northwestern Polytechnical University, Xi'an Shanxi 710072, China)

Abstract: This paper analyses the mechanism of two Linux's backdoor tools and points out their principles and harms. According to the characteristics of Linux, two kinds of protecting Linux's kernel methods are proposed in this paper, that is, single kernel module mode and LKM with security check mode. The realization method of Linux monitor based on LKM is also presented in this paper. The experimentations show that this monitor can efficiently record the attacks and illegal access from LKM backdoors and viruses, can help the system administrator to safeguard the security of Linux.

Key words: Linux; Loadable Kernel Module; Backdoor Tools; System Call

自从 Internet 出现以来, 安全问题就伴随着 Internet 的成长而发展。对于 Windows 操作系统来说, 它就是在与自身的 Bug、恶意病毒的斗争中不断地发展和完善自己。如今, Linux 操作系统由于其稳定、开放、高效等特点而得到广泛的应用, 特别是它提供的可装载内核模块 LKM (Loadable Kernel Modules) 机制, 使得程序员可以编写在内核级运行的代码而不必重新编译整个内核。当 LKM 被载入内核, 就能修改内核变量, 重载内核函数, 轻易地实现扩充或裁减操作系统内核的某些功能。然而, 这也为 Hacker 提供了可趁之机, 导致众多基于 LKM 机制的 Linux 后门工具和病毒程序的出现, 给 Linux 系统安全造成极大的威胁。

本文分析了 Linux 后门工具的实现机制, 设计实现一个基于 LKM 的 Linux 后门工具检测器。实验表明, 它能有效地记录 Hacker 对系统的攻击和非法访问。

1　LKM 后门工具的实现机制分析

LKM 后门之所以引起人们极大的关注, 是因为它利用现代操作系统模块技术作为内核的一部分运行, 这种后门比传统技术的后门更加不容易被发觉。一旦被安装运行到目标机器上, 系统就会完全由 Hacker 操纵, 管理员很难找到任何存在安全隐患的痕迹。

收稿日期: 2004-07-01; 修返日期: 2004-09-08
基金项目: 航空科学基金资助项目(01F53031); 教育部博士点基金资助项目(20020699026)

1.1　当前比较流行的 LKM 后门工具

当前较为流行的 LKM 后门工具有 Knark Rootkit 和 Adore Rootkit。Knark 是 Sekure.net 组织的成员 Creed 开发的一个基于 Linux 2.2 内核的后门工具, 它并不是一个稳定的版本, 在后面的开发中, 已经有针对 2.4 内核的版本出现。其主要功能是隐藏文件, 隐藏进程, 重定向可执行文件, 隐藏网络连接, 以 Root 身份运行命令等, 同时它为了入侵者查询方便, 把一些隐藏起来的文件、进程等信息放在/proc/knark 里。

而 Adore 是 Teso 小组成员开发的一个 LKM 后门工具, 其主要功能和 Knark 差不多, 不过它提供了卸载功能(需要提供密码验证), 支持 2.2 和 2.4 内核, 同时易于配置使用。

1.2　LKM 后门工具采用的主要技术

在用户空间执行每个命令都是调用内核的某些系统调用完成。比如用"ls"这个命令来查看当前目录信息, 就会调用 open(), gendents64(), write() 等系统调用。如果能截获 write() 这个调用, 修改它的输出, 那么现在的输出信息就不是真正的系统信息。LKM 后门最大的特点就是截获并修改多个系统调用, 从而改变整个系统响应。看一个截获 write() 系统调用的例子。

Netstat 命令用于查看网络连接状况, 显示命令结果会用到 write() 系统调用。在截获前, 命令 Netstat-at 的输出为

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:32768	*:*	LISTEN
tcp	0	0	*:ssh	*:*	LISTEN
tcp	0	136	192.168.0.253:telent	192.168.0.42:2056	ESTABLISHED

从输出中看出 IP 为"192.168.0.42"的用户通过 Telnet 连

接到了本机。现在利用一个 LKM 来截获 write( ) 系统调用,目的是把含有“192.168.0.42”这个字符串的信息屏蔽掉,从而实现隐藏网络连接的功能。关键代码如下:

```
int new_write( unsigned int fd, char * buf, unsigned int count )
{
  char * k_buf;
  char * hideinfo = "192.168.0.42";
  k_buf = (char * ) kmalloc(256, GFP_KERNEL);
  memset(k_buf, 0, 256);
  copy_from_user (k_buf, buf, 255);
  if ( strstr(k_buf, hideinfo) )
  {
  kfree(k_buf);
  return count;
  }
  kfree(k_buf);
  return real_write(fd, buf, count);
}
```

加载 LKM 后,write( ) 系统调用就被换成了 new_write( ),判断输出中是否含有“192.168.0.42”这个字符串,如果有,什么都不做,否则返回真正的 write( ) 系统调用。现在再来看命令 Netstat-at 的输出:

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	* :32768	* :*	LISTEN
tcp	0	0	* :ssh	* :*	LISTEN

成功隐藏! 再查看/etc/log 的登录记录,从“192.168.0.42”远程登录的记录也没有了。这就是 LKM 后门截获系统调用的威力。现在无论管理员用哪个命令,只要这个命令会用到 write( ) 系统调用,就不会显示含有“192.168.0.42”这个字符串的信息。这样既能隐藏当前的网络连接,又能擦除曾经登录而留在系统日志里的痕迹。所以在 Adore Rootkit 等后门中都实现了对 write( ) 系统调用的截获,隐藏 Hacker 的足迹。当然也可以用别的方式来修改这个输出,比如说把“192.168.0.42”改成“255.255.255.0”等任何你想修改成的 IP 地址来蒙骗管理员。

如果进一步截获其他的系统调用,这对系统管理员来说是致命的,这个操作系统将不再被信任。如截获 getdents64( ) 系统调用来隐藏文件和目录,截获 kill( ) 系统调用让管理员不能除掉木马进程,截获 getuid( ) 系统调用来设置一个能得到 Root 权限的账户等。

此外,还可以通过修改 query_module( ) 系统调用实现对模块自身的隐藏。这样,即使系统管理员发现系统有点不对劲,也不能从内核中发现后门的存在。

2 Linux 后门工具检测器的实现原理

2.1 检测器的设计思想

Linux 同 UNIX 一样,把结构分为两层,即内核和核外程序。Linux 后门工具要从核外程序发起对 Linux 内核的攻击和破坏,从机制上来说是不可能的,它唯一可利用的就是 LKM 方法。因此,Linux 后门工具检测器必须严守这道防线,保护系统内核不被攻击和窜改,这也正是该设计检测器程序的出发点。图 1 给出了检测器与 Linux 内核以及应用程序之间的层次关系。

根据用户的应用需求,本检测器提供两种保护内核方式。第一种方式可称作单模块内核方式,即系统在启动之后不允许任何人加载程序模块到内核中。这种办法虽然保证了内核的绝对安全,但同时对系统功能的扩充和开发也带来了不便。这种单模块内核方式适用于不准备进行任何系统扩充和开发的应用场合。第二种方式可称作带安全检测的 LKM 方式,这种方式适用于进行软件研发工作的应用环境。具体步骤是当系统启动完毕之后,首先为系统加载 LKM 检测器,由它来负责管理、监控随后所有的 LKM 活动。

图 1 检测器与 Linux 内核以及应用之间的关系

要加载一个模块,必须使用命令 Insmod 。我们利用 Strace 命令会发现它用到了 create_module( ) 这个系统调用,如果能截获 create_module( ) 系统调用,那么任何 LKM 的加载都将受到控制。对于单模块内核方式只要将 create_module( ) 修改成空函数就可以了;而对于带安全检测的 LKM 方式则要作一定的修改。

2.2 检测器的模块检测机制:记录每一个新加载的模块

利用 LKM 记录每一个新加载模块的名字。在调试 LKM 时有这样的经验:文件/proc/kmsg 记录了所有内核模块的 printk( ) 输出。可以这样考虑:截获 create_module( ) 系统调用,让它在加载每个 LKM 后将模块的名字记录在文件/root/log_lkm 中,那么内核模块的一举一动都在系统管理员的监视之下。下面是截获这个系统调用的关键代码:

```
int new_create_module( char * name, unsigned long size)
{
  char * k_buf;
  int ret = orig_create_module( name, size);
  k_buf = (char* ) kmalloc(256, GFP_KERNEL);
  copy_from_user( k_buf, name, 255);
  open = sys_call_table[ SYS_open ];
  write = sys_call_table[ SYS_write ];
  close = sys_call_table[ SYS_close ];
  int fd;
  char filename[ ] = "/root/log_lkm";
  mm_segment_t old_fs_value = get_fs( );
  set_fs( get_ds( ) );
  fd = open( filename, 0100 |02 |02000, 0640);
  write( fd, k_buf, sizeof( k_buf) );
  close( fd);
  set_fs( old_fs_value);
  return ret;
}
```

现在往内核里加载任意一个 LKM,在/root/log_lkm 里都记录下这个模块的名字。这样,系统管理员只需关注/root/log_lkm,就可以发现并删除任何非法的 LKM。之所以不直接输出到/proc/kmsg 中,是因为一个高明的 Hacker 也能通过查看/proc/kmsg 发现自己是否留下足迹,如果有记录则会通过别的方法擦除这个记录,毕竟他现在也有系统的最高权限。因此我们把这个记录放在别的不容易引起怀疑的地方,只要隐藏得好,Hacker 就找不到自己留下的脚印。

2.3 检测器的系统调用检测机制:记录系统调用地址的改变

LKM 后门工具的实现均是通过修改系统调用,这必然导致系统调用地址的改变。通过检测这些地址的改变情况,就能判断系统是否被加载了 LKM 后门。检测前先将未修改时的地

址保存起来(不妨保存在 log_lkm 中) , 然后利用另一个 LKM 来记录现在所有系统调用的地址, 为了对比方便, 把现在的系统调用地址存在/root 目录下的另一个文件 log_syscall 中。记录现在系统调用地址的 LKM 关键代码如下:

```
int init_module( void)
{
   open = sys_call_table[  SYS_open ] ;
   write = sys_call_table[  SYS_write ] ;
   close = sys_call_table[  SYS_close ] ;
   int j,  fd;
   char filename[  ]  = " /root/log_syscall" ;
   char addr[ 9] ;
   mm_segment_t old_fs_value = get_fs( )  ;
   set_fs( get_ds( ) ) ;
   fd = open( filename,  0100  | 02  | 02000,  0640) ;
   for ( j = 0;  j <240;  j + + )
   {
   sprintf( addr, " % x \n" , sys_call_table[ j] ) ;
   write( fd,  addr,  sizeof( addr) )  ;
   }
   close(  fd ) ;
   set_fs(  old_fs_value ) ;
   return 0;
}
```

将这两个 LKM 加载到内核后, 在 log_lkm 里记录了前 240 个系统调用最初的地址和后来加载的模块名字, 在 log_syscall 里记录了现在的系统调用的地址。通过对比, 如果某些系统调用的地址发生改变, 那么毫无疑问, 系统已经被 Hacker 加载了 LKM 后门。系统管理员现在应该分析文件 log_lkm 里新加载的模块记录, 找到是哪个 LKM 引起了系统调用地址的改变, 从而确定出 LKM 后门模块。

需要指出的是, LKM 后门工具往往只修改几个关键的系统调用。因此这个 LKM 可以简化为只检测几个重要系统调用的地址改变情况。这些重要的系统调用的是: getdents64( ) , kill( ) , read( ) , write( ) , fork( ) , clone( ) , execve( ) , getuid( ) , query_module( ) 。

2.4　检测器的隐藏机制

为了让这个检测器更安全、有效地运行, 还必须将它隐藏起来, 即隐藏模块本身和文件 log_lkm, log_syscall。隐藏文件的方法在后门工具 Knark 和 Adore 里都有介绍, 这里就不再赘述。现在主要实现如何在内核中隐藏模块。通过 Starc Lsmod 这个命令, 发现查询内核中的模块是利用 query_module( ) 这个系统调用。那就截获这个系统调用, 关键代码如下:

```
int new_query_module( const char *  name,  int which,  char *  buf,
size_t bufsize,  size_t *  ret)
{
   int r,  a;
   char *  ptr,  *  match;
   r = real_query_module(  name,  which,  buf,  bufsize,  ret ) ;
   ptr = buf;
   for ( a = 0;  a  < *  ret ;  a + + )
   {
      if ( ! strcmp(  LKM_NAME,  ptr) )
      {
         match = ptr;
         while ( *  ptr)  ptr + + ;
         ptr + + ;
         memcpy( match,  ptr,  bufsize-( ptr-( char * )  buf) ) ;
         ( *  ret)  --;
         return r;
      }
      while ( *  ptr)  ptr + + ;
      ptr + + ;
   }
   return r;
}
```

2.5　检测器的加载机制

为了让检测器始终在内核中运行, 不会因为系统重启而被 “踢出 ”内核, 管理员必须把载入这个检测器的行为放到系统的启动序列中。简单的方法是在/etc/rc. local 中加入一行: Insmod LKM. o, 来实现每次系统启动都会加载这个 LKM 检测器。

3　结束语

通常的检测器都是针对某个特定的命令, 如 Chkrootkit 就是通过检测 “ls ”和 “ps ”的输出与/proc 下的目录是否相符来作出判断, 对其他高明的后门如 Knark, Adore 等就束手无策。本文针对 LKM 后门工具和病毒的普遍实现机制而提出了一种检测方法, 可以有效地检测各种后门的加载情况, 从而极大地提高了系统的安全性。

参考文献:

[1]　ragmatic. Complete Linux Loadable Kernel Modules version 1. 0 [EB/OL] . http: //www. xfocus. net/articles/200008/47. html, 1999.

[2]　Stealth. Kernel Rootkit Experiences[EB/OL] . http: //www. xfocus. net/articles, 2003.

[3]　范磊. Linux 内核源代码[M] . 北京: 人民邮电出版社, 2002.

[4]　王永杰, 刘京菊, 孙乐昌. Linux 可装载模块的开发与应用[J] . 计算机应用研究, 2002, 19(7) : 143-146.

[5]　宋立新, 李善平. 利用 LKM 提高 Linux 系统的安全性[J] . 计算机应用研究, 2000, 17(8) : 103-105.

[6]　丁志芳, 徐孟春, 李晓秋, 等. Linux 安全模块的设计与实现[J] . 计算机应用, 2003, 23(6) : 289-291.

作者简介:

袁源(1979-) , 男, 硕士研究生, 主要研究方向为计算机网络、网络信息安全等; 罗红(1963-) , 男, 副教授, 博士研究生, 主要研究方向为网络安全、网格计算等; 戴冠中(1937-) , 男, 教授, 博士生导师, 主要研究方向为自动控制、信息安全; 吕鹏(1979-) , 男, 硕士研究生, 主要研究方向为计算机网络、网络信息安全等。