基于VMM的文件完整性监控系统的设计与实现

陈 威[1] 王 晖[2]
(北京航空航天大学中法工程师学院 北京 100191)[1] (首都经济贸易大学 北京 100070)[2]

摘 要 虚拟机监控器(VMM)具有强控制性、隔离性的特点。针对现有文件完整性监控系统中存在的缺陷,提出了一种新的基于VMM且与客户机相隔离的文件完整性保护方法,该方法能够保护用户的敏感文件,特别是文件完整性监控系统本身,使其免受恶意代码的攻击。这种基于虚拟机监控器的文件完整性保护解决方案,在虚拟机隔离层中通过设计和嵌入的"探测器"和"文件逆向定位器"两种关键技术,能够实时地探测到对被保护文件的所有访问企图,从而实现预置的保护策略。

关键词 虚拟化技术,虚拟机,I/O截获,文件完整性保护
中图法分类号 TP302 **文献标识码** A

Design and Implementation of VMM-based File Integrity Monitoring System

CHEN Wei[1] WANG Hui[2]
(College of Sino-French Engineer, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)[1]
(Capital University of Economics and Business, Beijing 100070, China)[2]

Abstract A virtual machine monitor(VMM) has strong control ability and its characteristic of isolation, and can solve open question in the existing file integrity monitoring systems. A new VMM-based method for file integrity protecting system was proposed, which is isolated between the system and the guest systems. This method should pre-configure the files to be protected and can avoid the attack to these files from the malicious codes. In this scheme of file integrity protection, the system can intercept all the access attempts to the protected files in real-time by designing and implanting the "detector" and "reversed file locator" into the isolated layer of the virtual machine, and achieves the strategy of pre-protection.

Keywords Virtualization technology, Virtual machine, I/O interception, File integrity protection

1 问题的提出

随着计算机硬件及其技术的发展,虚拟化技术在科研、商务领域都得到了愈加广泛的应用。但是,虚拟机同主机一样,都面临着计算机病毒、木马等恶意代码的威胁,这些恶意代码的主要攻击目标就是系统中的文件。目前针对虚拟机内文件完整性的保护工具主要有两种:基于主机的文件完整性监控系统以及基于虚拟机的文件完整性监控系统。然而,这两种系统均存在着被恶意代码感知、绕过甚至是破坏的危险。下面进行分析。

1)基于主机的文件完整性监控系统。以其代表性系统TripWire为例,工作原理见图1。

TripWire的核心技术就是对每个要监控的文件产生数字签名,当文件的数字签名与数据库中保留的数据不一致时,就会做出文件被篡改的判断,并将结果报告给管理员。该系统的缺陷在于数据库和恶意代码运行于同一环境中,曝露在恶意代码的攻击范围内,如果被恶意代码攻击,该系统将不再可靠[1,2]。

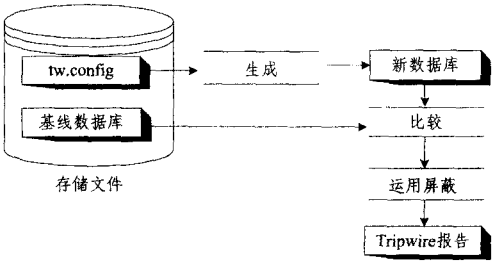

图1 TripWire工作原理

2)基于虚拟机完整性监控系统。以其代表性系统XenFIT为例,工作原理见图2。

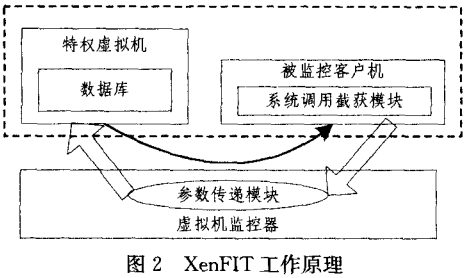

图2 XenFIT工作原理

到稿日期:2012-05-17 返修日期:2012-06-12 本文受国家科技支撑计划项目(2010BAH57B03)资助。
陈 威(1986-),男,硕士生,主要研究方向为计算机模式识别、计算机视觉,E-mail:Wei.CHEN.LIRIS@gmail.com;王 晖(1985-),男,博士,主要研究方向为医院信息化、计算机网络,E-mail:cthcth123@126.com(通信作者)。

XenFIT 是在被监控系统中嵌入了一个用以截获系统调用的模块，并将数据库放置在一个特权虚拟机中。其优点是数据库处于与恶意代码相隔离的环境中，安全得以保障，但用以截获系统调用的模块却与恶意代码运行于同一环境，容易被攻击或绕过[3,4]。

以上两种文件完整性监控系统存在的缺点就是系统自身处于恶意代码的攻击范围内，自身的安全无法得到保证。因此，本课题设计的基于虚拟机监控器的文件完整性监控系统解决方案，除保护预置保护的文件的安全外，还要避免保护系统本身不会被恶意代码攻击。

2 系统设计

2.1 总体设计

虚拟化技术的核心是一个运行在硬件层之上的虚拟机监控器。这个虚拟机监控器会在虚拟服务器和底层硬件之间建立一个抽象层，并且能够捕获客户机的 CPU 指令，为指令访问硬件控制器和外设充当一个访问的中介，因而，拥有极强的隔离性和干涉性。虚拟化技术也为每一个在它之上运行的虚拟机提供一个隔离的环境，这些虚拟机之间无法相互感知、影响，并且这些虚拟机无法感知到虚拟机监控器层的存在，无法对虚拟机监控器造成任何的影响或修改，但是虚拟机监控器却能够完全地控制、干涉上层虚拟机的行为。

虚拟化技术为解决本课题的问题提供了一个有启发性的思路。基于虚拟机的体系结构，并结合其特有的强隔离性、干涉性以及控制性，提出了一种新的、具有与客户机相隔离的文件完整性保护方法。该方法能够对所要保护的文件预先设置保护，并保证文件完整性监控系统本身免受恶意代码的攻击。

系统总体结构设计如图 3 所示。

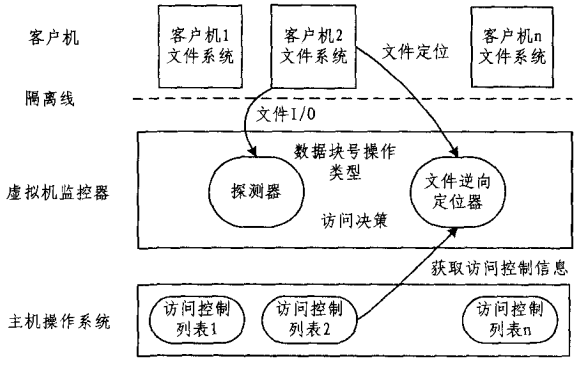

图 3 系统总体结构图

图 3 中的虚线称作隔离线，表明了客户机 1 至客户机 n 可以感知的范围，其处于独立的隔离环境中，无法向下感知到虚拟机监控器的存在。并且客户机之间也无法相互感知和影响。但是，虚拟机监控器却能够完全地控制上层客户机的行为。

基于以上特点，将所需功能嵌入到虚拟机监控器中，将访问控制列表放置在宿主机中，这样虚拟机监控器中的监控系统和访问控制列表都处于客户机内恶意代码无法感知的下层，因此监控系统以及访问控制列表自身的安全性有效地得到了保证。

每一个客户机都对应着唯一的访问控制列表，访问控制列表中记录需要保护的文件路径、文件名和保护的等级。本系统提供了两个保护等级：只读和不可访问。对于客户机中未被设置保护的文件，系统将采取默认访问权限，即不对此类文件读写操作进行处理，而是客户操作系统根据内部的权限自行处理。

基于上述对系统的总体设计，系统监控流程的设计如下。

在虚拟机启动时，监控系统会首先从宿主机中读取对应客户机的访问控制列表，对列表中的文件进行解析后，将文件的信息存放至相应的数据结构内。当客户机发出一个 I/O 请求后，虚拟机监控器通过陷阱捕获此 I/O 请求，并会根据该请求是读、写进入不同的分支。在此，系统将此 I/O 请求所操作的扇区号解析并转换为数据块号。在得到了数据块号后，系统会根据数据块号逆向定位查找到所属的文件信息，并根据操作类型对比访问控制列表判断本次操作是否允许，如果不，系统就会加以阻止。

本文所设计的文件完整性监控系统是基于 qemu-kvm-0.11.0 的源代码完成的。QEMU(Quick EMUlator)是由 Fabrice Bellard 所编写的模拟处理器的自由软件，具有高速、跨平台的特点。KVM(Kernel-based Virtual Machine)是一种基于硬件的完全虚拟化的虚拟机监控器，它需要 CPU 支持虚拟化技术。KVM 充分地利用了 Linux 的内核，简化了管理并使得性能得到了提升。

2.2 探测器设计

探测器设计的主要作用是，捕获对文件的 I/O 访问企图，并从中鉴别出对被保护文件的每一个 I/O 及其访问类型，依据预置的保护策略进行相应的处理。主要功能包括两个方面。

2.2.1 I/O 截获模块

QEMU_KVM 的 I/O 模型如图 4 所示，KVM 主要负责内核部分，而硬件等装置均由 QEMU 进行模拟，能用来在一部主机上虚拟出数部不同的虚拟计算机。

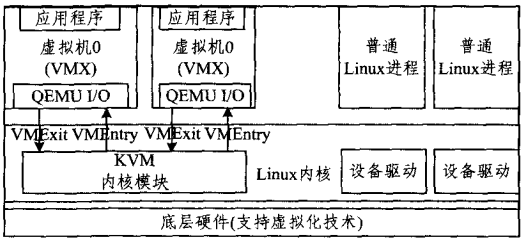

图 4 QEMU 的 I/O 结构示意图

当应用程序发起一个 I/O 指令时，由于 I/O 模型直接来自于 QEMU，虚拟机监控器能够通过陷阱捕获 I/O 指令，然后调度并转发到特权客户，调用特权客户中断处理程序，执行进程上下文切换，通过 I/O 进程初始化客户的 I/O。I/O 初始化结束之后，虚拟机监控器切回到最初的客户，继续执行客户代码。

通过分析 qem-kvm-0.11.0 源代码可知，当虚拟机发出读写的 I/O 请求时，会产生 I/O 中断陷入到虚拟机监控器中，然后判断中断的类型(读或写)，从而选择进入不同的分支。因此，I/O 截获模块的探头可以设置于读、写两个分支中。由不同的分支，自然也就获取了文件访问读或写的类型，同时，利用 ide_get_sectors(IDEState * s)函数能够将截获的 I/O 请

求中操作的扇区号解析出来，从而扑获到 I/O 访问目标。

EXT2(Second Extended File System)是 Linux 内核所使用的文件系统之一。在 EXT2 文件系统中，所有的数据结构大小均基于块，而非扇区。一个 EXT2 文件系统中的每个块的大小是一样的，但是对于不同的 EXT2 文件系统，块的大小是可以不同。最常用的块大小为 1024 Bytes 或者 4096 Bytes。块的大小在 EXT2 文件系统建立时就被决定，它由管理员选择，也可以由文件系统的创建程序根据硬盘分区的大小自动选择。因此，若要解决每一个 I/O 与文件的关联，需要将扇区号转换成文件块号，以便进一步逆向定位所要访问的文件。而且，不同的 EXT2 文件系统块的大小可能不同，因此，这个转换也是动态的。扇区与块号转换的方法如下：

Block=sector * (unit / sector size)

至此，探测模块已经获得了本次 I/O 所要访问的数据块号和访问类型。

2.2.2 I/O 访问的处理

当探测器扑获到本次 I/O 后，需要进一步判断本次访问的目标文件是否在预置保护的列表中。这个判断在文件逆向定位器模块中完成，在下一节中介绍。依据逆向定位器模块返回的结果，决定对本次 I/O 的处理：阻止或放行。

在 qemu-kvm-0.11.0 源代码根目录的 qemu-common.h 文件中，有两个重要的向量结构体，这两个结构体记录了本次 I/O 操作的基地址以及数据长度。结构体如下：

QEMUIOVector 结构体：

```
struct QEMUIOVector {
    struct iovec *iov;
    int niov;
    int nalloc;
    size_t size;
}
```

iovec 结构体：

```
struct iovec {
      void *iov_base;
      size_t iov_len;
}
```

这两个结构体对 I/O 操作的写入和读取有着重要的作用。系统若要阻止某次 I/O 操作，将本次 I/O 操作的写入数据长度设置为 0 即可。

系统设计了 searchBlocks(int operation, int64_t blockNum)函数，其作用是将转换成的数据块号以及此次读、写操作类型作为参数传递给文件逆向定位模块，在模块返回了该块号是否属于受到保护的数据块的信息之后，便会根据所设定的保护等级来进行处理。

对本次 I/O 有 3 种处理形式：

本次 I/O 所访问的块号不在被保护文件对应的块号列表中，一律放行；

本次 I/O 是读访问，访问的块号所对应的文件是：只读访问，放行；拒绝访问，阻止。

本次 I/O 是写访问，访问的块号所对应的文件是：只读，阻止；拒绝：阻止。

2.3 文件逆向定位器设计

由前述可知，在探测器中，可以获得一个 I/O 所要访问的数据块号和访问类型。要防止对预置的保护文件的非允许访问，首先要知道本次 I/O 访问的数据块是否定位在被保护文件所存储的数据块列表中以及对该数据块的操作类型，然后决定对本次 I/O 的处理。

所谓逆向定位，是指由数据块号反溯确定是否是被保护的文件，系统将牺牲比较大的效率。一种比较优化的办法是，系统初始化时，事先将被保护文件所对应的文件块号及保护类型解析出来并进行排序，对每个 I/O，与此数据块列表进行比对即可。

2.3.1 虚拟机镜像文件解析

在 EXT2 文件系统中，文件由 inode 进行唯一标识，它包含文件的所有信息。EXT2 文件系统采用三级间接块来存储数据块指针，并以块为单位分配空间。而对于每一个索引节点，其大小均为 0x80 字节，也就是 128 字节，其中记录了文件在分区里面的位置，这些有关位置的数据均是以块为单位，该索引节点所指向的第一个块号被记录在索引节点的第 41—第 44 字节中，并且在第 45—第 100 字节中，记录了其他的 14 个数据块的指针。

需要提到的是，在总共 15 个指针中，前 12 个即第 41—第 88 字节的指针被称为直接指针，这 12 个指针直接指向属于该节点的数据块号；第 89—第 92 字节被称为一次间接指针，这个指针并不是直接指向数据块号，而是指向一个用以记录数据块号的数据块；第 93—第 96 字节被称为二次间接指针，同理，它指向的数据块同时又指向了新的数据块，而这些数据块中才记录了用来存放该索引节点信息的数据块号；以此类推，第 97—第 100 字节被称为三次间接指针。一个 128 字节的索引节点结构如图 5 所示。

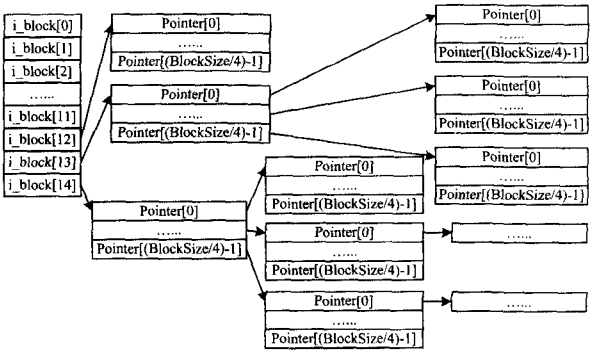

图 5 索引节点中直接指针及间接指针示意图

由此，通过索引节点中的直接指针、一次间接指针、二次间接指针、三次间接指针，即可设计程序来获得一个文件所有的数据块号。

为了提高系统处理的性能，我们对获得的所有文件所占用的块号进行“快速排序(QuickSort)算法”处理，采用一种分治的策略将给定的一个无序数列快速转换为从小到大的有序数列。

在得到了数据块后，理论上就可以对解析出的 I/O 请求所涉及的数据块号进行比对，判断所请求操作的数据块号是否属于被保护的文件。由于很可能会出现需要保护的数据非常大的情况，因此不宜使用遍历的方法。这里采用的是二分查找(Binary Search)的方法。二分查找要求数据是有序的，并且需要使用向量作为表的存储结构。

2.3.2 索引节点的获得

EXT2(Second Extended File System)文件系统由一系列被称为块组(Block Group)的逻辑结构组成,见图 6。

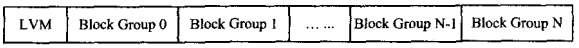

图 6 EXT2 文件系统布局

而一个块组由超级块、描述符、块位图、节点位图、节点表和数据块组成,见图 7。

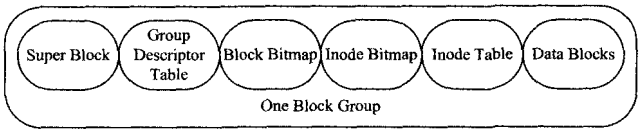

图 7 EXT2 文件系统块组结构图

在 EXT2 文件系统的结构中,使用了一个叫 inode 的数据结构来描述文件系统中某个文件或目录的索引节点。每个文件都由一个拥有唯一号码的 inode 来描述,这个 inode 描述了每个文件的位置、大小、权限、修改创建时间、文件类型等。简单来说,索引节点就是一个文件的钥匙,当需要对文件进行访问时,最重要的是获得该文件的索引节点的首地址。

获取索引节点首地址的方法如下:

1. 获取虚拟机镜像的偏移量:在系统启动的过程中,利用"fdisk -l"命令,可以获得虚拟系统镜像信息,经过解析得到包括该虚拟机磁盘的第一分区的起始扇区数和单位(units)大小。由此,可以计算得到虚拟机镜像的偏移量:

Offset=(start unit) * units。

2. 获取受保护文件的索引节点号(inodeNum):Linux 提供了"ls -i"命令,可以得到文件的索引节点号。通过该命令可以将文件的索引节点号保存至目标文件中。但是,由于受保护的文件存在于虚拟机镜像文件中,在主机环境中并不能利用该命令获得受保护文件的节点号。采用的方法是,首先将虚拟机镜像挂载,使受保护文件在主机下可见,然后读取访问控制列表中的文件路径和名称,使用"ls -i"逐一获得对应的索引节点号。挂载命令如下:

mount -o loop,offset=offval (其中 offval 为第 1 步中得到的偏移量)

3. 从超级块(Super Block)中获得相关参数:每个 EXT2 文件系统包含有一个超级块,超级块的起始位置是它所在分区的第 1024 个字节,它占用 1kB 的空间,由结构体 struce ext2_super_block{}进行定义。根据虚拟机镜像偏移 offset 可以读得这个超级块,并根据结构体的定义,解析其中的第 40—第 43、第 88—第 89 字节中保存的整数即可得到 inodesPerGroup、inodeSize,第 24—第 27 字节的整数作为 2 的指数得到 blockSize。

4. 从块组描述符(Group Descriptor Table)中获得相关参数:同样,根据虚拟机镜像偏移 offset,在超级块之后读得块组描述符,块组描述符由若干块组描述符组成,由结构体 struce ext2_group_desc{}进行定义。按照每个块组描述符 32 个字节,提取相应的块组描述符,之后再解析出第 8—第 11 字节保存的 gdts[groupNum]。

5. 计算索引节点首地址:计算索引节点首地址的公式如下。

groupNum=[(inodeNum-1) / inodesPerGroup] 方括号表示向下取整

inodeOffset=(inodeNum-1) % inodesPerGroup 百分号表示求余数

inodeAddr=gdts[groupNum] * blockSize+inodeOffset * inodeSize

将前几步得到的参数带入上述公式即可得到索引节点的首地址。

3 实验和结论

3.1 实验环境

基于虚拟机监控器的文件完整性监控系统的设计是基于以下宿主机环境实现的:

- CPU:Intel(R) Core(TM)2 CPU 2.63GHz,支持虚拟化技术
- 内存:2.72G
- GCC 版本:gcc 4.6.1 (Ubuntu/Linaro 4.6.1-9ubuntu3)—7.3-2011.08
- GDB 版本:GNU gdb (Ubuntu/Linaro 7.3-0ubuntu3)—7.3-2011.08
- 文件系统:EXT2

3.2 功能实验

实验目的是验证嵌入本系统的 qemu-kvm-0.11.0 是否可以保护用户敏感文件不受非法破坏。

首先制作虚拟机镜像,并安装操作系统 Debian,然后在宿主机内安装修改后的 qemu-kvm 虚拟机。在虚拟机的/root/目录下建立了两个文本文件:aaa.txt 和 bbb.txt,作为被保护的测试文件。通过敏感文件列表设置功能,启动虚拟机之前用户可以添加、删除和修改需要保护的文件以及保护类别:0 为可读,1 为不可访问。此次测试将文件 aaa.txt 的保护类别设置为 1;文件 bbb.txt 的保护类别设置为 0。启动虚拟机,其保护即可生效。

读测试的程序源文件(read_test.c)如下:

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc,char * * argv) {
  char text[1024];
  FILE * fp;
  fp=fopen(argv[1],"r");
  fgets(text,1024,fp);
  printf("%s\n",text);
  fclose(fp);
  return EXIT_SUCCESS;
}
```

写入测试的程序源文件(write_test.c)如下:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc,char * * argv) {
  FILE * fp;
  fp=fopen(argv[1],"a");
  fwrite(argv[2],1,strlen(argv[2]),fp);
  fclose(fp);
```

```
return EXIT_SUCCESS;
}
```

为了方便测试，修改后的 qemu-kvm 会对虚拟机初始化和对虚拟机被保护文件的访问做出反应，实时地在宿主机屏幕上显示有关信息。

测试结果如下：

1. 虚拟机初始化测试

当一个虚拟机启动时，宿主机终端显示图 8 信息。

图 8　初始化后主机终端截图

从图 8 可以看到，初始化时，系统已经获得了实施文件定位的关键信息，包括系统目录、虚拟机镜像名称、虚拟机镜像路径、虚拟机分区偏移量、超级块信息、受保护文件数量、受保护文件信息等，通过这些信息及前述算法，可以圈定被保护文件的块列表。

2. 对拒绝访问的文件进行读取测试

测试命令为：read_test aaa. txt

宿主机终端显示信息如图 9 所示。

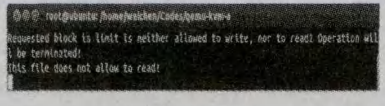

图 9　主机终端截图，读取操作被终止

可以看到，对 aaa. txt 的读取被阻止，结果正确。

3. 对拒绝访问的文件进行写入操作

测试命令为：write_test aaa. txt test

宿主机终端显示信息如图 10 所示。

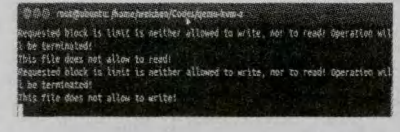

图 10　宿主机终端显示，写入操作被终止

可以看到，对 aaa. txt 的写入被阻止，结果正确。

4. 对只读文件进行读取的操作：

测试命令为：read_test bbb. txt

虚拟机终端显示信息如图 11 所示。

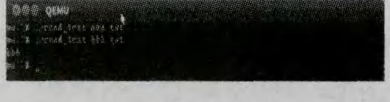

图 11　对 bbb. txt 文件进行读取测试，内容正常显示

因为 bbb. txt 文件允许读，系统对本次读取操作并不进行阻止，所以程序能够正常地显示出 bbb. txt 文件的内容。

5. 对只读文件进行写入的操作

测试命令为：write_test bbb. txt test

宿主机终端显示信息如图 12 所示。

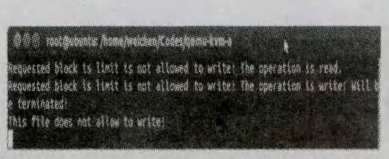

图 12　主机终端截图，写入操作被终止

可以看到，对 bbb. txt 的写入被阻止，结果正确。

通过上述实验以及截图可知，修改过的 qemu-kvm-0. 11. 0 可以保护该虚拟机用户指定的文件不受到越权操作，若文件为只读文件，则任何对该文件的写操作都不会起作用；若文件被指定为不可读也不可写，则任何对该文件的读写操作均无效。

3.3　性能实验

qemu-kvm-0. 11. 0 源代码嵌入本系统后对性能会造成一定的影响，其主要原因是系统对每个截获的 I/O 所解析出的数据块要与被保护文件的数据块列表进行比对。

Iozone 是一个文件系统的 benchmark 工具，可以测试不同的操作系统中文件系统的读写性能。本实验是在虚拟机中安装 iozone 程序，并利用该程序对虚拟机进行读、写测试，测试被保护文件的大小、读写文件的大小以及每次读写数据的大小，与未嵌入本系统时进行对比，测试对读写速度的影响。

具体方法是利用命令：

./iozone -a -n 1m -g 128m -i 0 -i 1 -f /tmp/iozone /root/iozone. xls

该命令对文件系统作用的环境设置是：

1. 被保护文件的大小。分为未安装保护和被保护文件大小为 1M，2M，4M，10M，20M，40M 等 7 种状态。

2. 进行读写的文件大小。从 1M 到 128M，间隔为 2 的倍数。

3. 每次写入的数据大小。从 4k 到 16M、间隔为 2 的倍数。

上述测试共进行了 10 次，众多的测试数据的分析结果显示，当测试文件大小一定时，读取或写入速度的变换与被保护文件大小的变化形态基本一致。下面撷取测试文件为 16M，每次写入的数据大小为 4k 时，读写速度的变化情况，如图 13、图 14 所示。

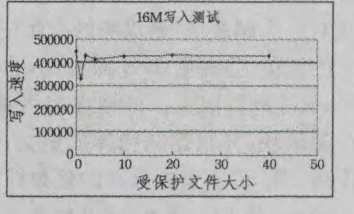

图 13　16M 测试文件，4k 记录大小写入测试

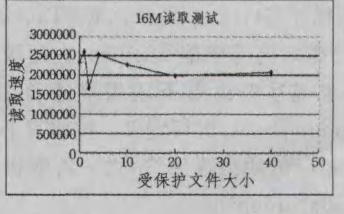

图 14　16M 测试文件，4k 记录大小读取测试

（下转第 265 页）

和最大的5个分类正确率所对应的权值系数，其中表3是采用全局随机法的情况，表4是采用局部随机法的情况，std表示权值系数的标准差，W是最终分类正确率。从表3、表4可以看出：分类正确率较小的权值系数标准差过大(大于0.01)或过小(小于0.005)，而分类正确率较大的权值系数标准差集中在区间(0.005,0.01)内，当$L=3,7,9$时此规律也基本符合。也就是说，权值系数既不能过于集中也不能过于分散，必须在一定的离散范围内。其原因在于，权值系数是根据每个加权分量对测试数据的分类正确率决定的，权值的大小和分类正确率成正比关系。权值系数过于集中，则说明所有加权分量的分类正确率都比较接近，而相近分类正确率在"投票策略"中贡献相近，可以认为只有一份贡献，变相地降低了加权分量的个数；权值系数过于分散，则说明有一个加权分量的分类正确率过低，它在"投票"过程中所能起的作用十分有限，甚至可能起负作用。

结束语 核Fisher判别分析法是一种有效的提取非线性特征的方法。本文提出一种加权多核Fisher判别分析方法。所提方法通过将多个由单核Fisher得到的投影映射进行加权组合，得到一个加权投影映射，以加权投影来提取测试数据的特征并进行分类。实验结果表明，加权多核Fisher判别具有优于单核Fisher判别的分类正确率。

在加权多核Fisher判别的实现中，核参数系数组合的选择是关键。本文中，在限定的不重叠区间内随机选择核参数系数进行加权多核Fisher判别得到的分类正确率有很大几率高于单核Fisher判别的最高分类正确率。是否能够进一步细化限定区间，使得从细化的限定区间内随机选择核参数系数进行加权多核Fisher判别得到的分类正确率一定高于单核Fisher的最高分类正确率，是要进一步研究的问题。另外，选择不同形式的核函数计算投影分量能否提高分类正确率，也是需要进一步研究的问题。

参考文献

[1] 葛微，程宇奇，刘春香，等. 基于子空间的人脸识别算法研究[J]. 中国光学和应用光学，2009，2(5)：377-387

[2] 边肇祺，张学工. 模式识别(第2版)[M]. 北京：清华大学出版社，2000

[3] Mika S，Ratsch G，Weston J，et al. Fisher discriminant analysis with kernels[C]//Proceedings of IEEE International Workshop on Neural Networks for Signal Processing. Madison，Wisconsin：IEEE，1999：41-48

[4] 汪洪桥，张富春，蔡艳宁，等. 多核学习方法[J]. 自动化学报，2010，36(8)：1037-1050

[5] John S T，Nnello C. Kerenl methods for pattern anlaysis[M]. Beijing：China Machine Press，2005

[6] 李映，焦李成. 基于核Fisher判别分析的目标识别[J]. 西安电子科技大学学报，2003，30(2)：180-182

[7] MIT-CBC. MIT-CBCL Face Data [DB/OL]. http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html，2000

(上接第256页)

图14中横坐表为0点的速度是未嵌入本系统时的测试文件写入速度。可以看出，写入速度并不完全与被保护文件的大小呈简单关系，反复测试结果与图示大体是一致的。但总体上写入速度随被保护文件的增大而降低，上述测试点的平均速度约降低了4%，对用户影响不大。

横坐表为0点的速度是未嵌入本系统时的测试文件的读取速度。与写入时情况有些类似，读取速度也不是与被保护文件的大小呈简单关系，反复测试结果与图示大体是一致的。但总体上读取速度随被保护文件的增大而降低，上述测试点的平均速度约降低10%左右，数据反映出其对系统具有一定的影响，但在用户使用中感觉并不明显。

对于读写速度未呈现设想的变化，可能的原因是虚拟机所建造的环境是基于宿主机之上的，非完全预期的结果尚不能确定是否来自于虚拟机的运行机制以及iozone测试工具的原因。但总体的测试数据显示基于虚拟机监控器的文件完整性监控系统对性能的影响在可以接受的范围之内。

结束语 本文提出了一种基于虚拟机监控器的文件完整性监控方法，并设计了代码来加以实现。通过实验可知，该系统能够对用户所配置的访问控制列表中的文件提供预期的、可靠的保护，并且解决了基于主机的文件完整性监控系统和基于虚拟机的完整性监控系统自身受到恶意代码攻击的问题，是一种为用户重要的敏感文件提供安全保护的可靠工具。其缺点是，对新加入的保护文件，系统需要重新启动才能生效，这是系统下一步需要改进的地方。同时，系统性能上也可以进一步进行优化。

参考文献

[1] 冯力. TripWire在入侵检测系统中的应用[J]. 信息安全与通信保密，2002(5)

[2] Kim G H，Spafford E H. The design and implementation of tripwire：A file systemintegrity checker[C]//2nd ACM Conference on Computer and CommunicationsSecurity. Fairfax：ACM，1994：18-29

[3] Quynh N A，TakeFuji Y. A real-time integrity monitor for Xen virtual machine[C]//Proceedings of the 1st ACM Workshop on Virtual Machine Security. New York：ACM，2009：1-10

[4] Quynh N A，Takefuji Y. A novel approach for a file-system integrity monitor tool of Xen virtual machine[C]//2nd ACM Symposium on Information，Computer and Communications Security. Singapore：ACM，2007：194-203

[5] Jin Hai，Xiang Guo-fu，Zou Deq-ing，et al. A guest-transparent-file integrity monitoring method in virtualization environment [C]//Computers andMathematics with Applications. 2010：18-29

[6] Riley R，Jiang X，Xu D. Guest-transparent prevention of kernel rootkits with VMM-based memory shadowing[C]//11th International Symposium on Recent15 Advances in Intrusion Detection. Massachusetts：Springer，2008：1-20

[7] Payne B D，Carbone M，Sharif M，et al. Lares：An architecture for secureactive monitoring using virtualization[C]//2008 IEEE Symposium on Securityand Privacy. California：IEEE，2008：233-247