强制访问控制在 Windows 上实施框架的研究与改进①

陈 徽 周学海 陈香兰 (中国科学技术大学 计算机科学与技术系 安徽 合肥 230027; 苏州市嵌入式系统重点实验室 江苏 苏州 215123)

摘 要: 计算机技术发展至今，安全问题一直处于风头浪尖之中。目前针对安全问题，一些技术方法应运而生，主动防御，侦测与反侦测等，而且可以在系统中不同的层次上实现，应用层和内核层。而本文是在系统级别上进行探索，比如 linux，现在已有成熟的安全版本 selinux 操作系统。针对 Windows 操作系统，对强制访问控制进行了相关研究，对系统的效率性，兼容性，稳定性等提出了改进方案。

关键词: Hook；强制访问控制；Windows；系统安全；系统服务描述表

Study and Improvement of MAC(Mandatory Access Control) for Windows

CHEN Hui, ZHOU Xue-Hai, CHEN Xiang-Lan

(Department of Computer Science, University of Science and Technology of China, Hefei 230027, China; Suzhou Key Lab for Embedded System, Suzhou 215123, China)

Abstract: Security is always a hot topic with the development of computer technologies. Currently, mechanisms like active defense, detection and anti-detection are proposed. They could be implemented on different levels such as application-level and kernel-level in the system. We manage to go to the system-level. Linux owns mature security-enhanced version SELinux. For Windows, we focus on mandatory access control and propose a mechanism with improved efficiency, compatibility and stability.

Keywords: Hook; MAC; Windows; system security; SSDT

1 引言

目前，Windows 操作系统是一款非常流行的桌面操作系统，同时也是安全防护最关注的操作系统之一[1]。在目前广泛流行的防火墙技术，杀毒软件的情形下，安全操作系统的研究也变得日益受到人们的关注。自主访问控制由于安全粒度比较大，而且它本身的一些缺陷，导致了操作系统很容易被暴力破解，字典攻击等手段攻破[2]。强制访问控制的研究使得安全粒度更加的细化，使得安全操作系统成为我们的研究重点。

目前 Linux 已经有成熟的安全操作系统版本 Selinux[3]，对系统中的安全对象添加了安全域，对它们的关键操作实施了拦截控制。这两个基本点和庞大的策略库是 Selinux 的基石。Selinux 实施的安全策略机制基于强制访问控制机制，而且提供了可配置的安全策略[4]。针对开源的 Linux 操作系统，安全框架的实施比较直观明了，然而对 Windows 操作系统而言，因为其核心代码没有公开，我们只能采取其提供

① 基金项目:电子信息产业发展基金(财建[2008]329,工信部运[2008]97)
收稿时间:2010-03-21;收到修改稿时间:2010-04-08

的技术手段。

Windows 系统访问控制实施框架 WAEF[5]是针对 Windows 操作系统提出的一个设计框架。它是基于 Windows Hook 技术对系统的对象进程、文件、注册表以及网络套接字进行安全监控。然而在 Windows 操作系统上实现对目标和操作的监控，对性能和稳定性有着很大的影响。通过分析和研究 Windows 操作系统本身特性以及强制访问控制在 WAEF 上的实施，本文提出一个加强版的访问控制框架 Windows 系统访问控制实施框架 (Windows Enhanced Access- control Enforcement Facility Framework，WEAEF)。通过分析影响性能和稳定性的几个方面来进行设计的改进，如进程、文件、网络套接字以及策略库的访问。

文章第二节阐述 WAEF 的框架设计，重点剖析影响 Windows 操作系统性能和稳定性的几个方面。在第三节提出目前存在的问题，以及针对这些问题提出自己的设计思路。第四节对第三节提出的改进思路设计了实现方案。在第五节中，对不同的实验方案进行比较，得出新提出的设计方案的优势。最后，为了进一步的研究，我们进行总结以及提出以后的工作重点。

2　WAEF框架概述

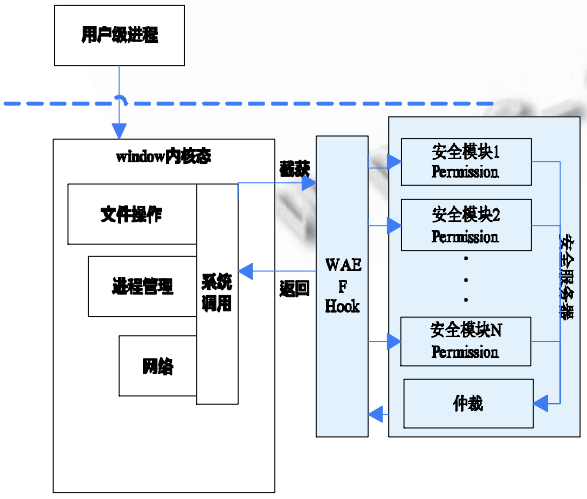

图 1　WAEF 多安全模块支持[5]

WAEF 是适用于 Windows NT 系列的安全访问控制设计框架。它实现了进程、文件、注册表和网络部分 SOCKET 的访问控制，如图 1。

WAEF 是基于成熟的 HOOK 技术，通过修改 SSDT 表来对系统内核层的对象进行控制。系统上层的一些操作，比如：打开一个文件，如图 2。从用户在应用层执行这个操作起，在内核层调用系统 API 之前，WAEF 进行了拦截控制。成功拦截之后，WAEF 首先根据自己获得相关信息(文件名、安全属性等等)，访问策略库进行判断，如果允许则调用原来的系统 API，负责返回拒绝，记录日志信息。

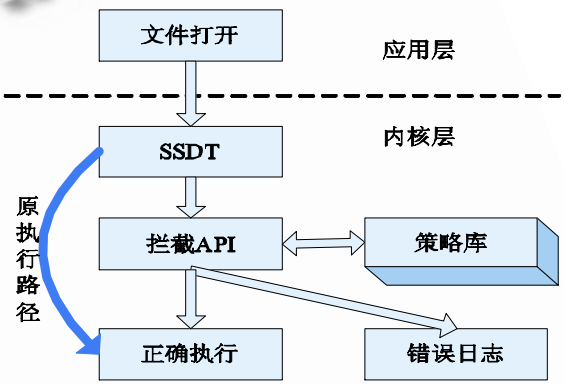

图 2　WAEF 文件打开框架

关于进程的设计，WAEF 利用进程描述符的空闲域，采用添加安全项的方法存储进程的安全描述符，而进程描述符的空闲域的偏移量不是固定的；关于文件的设计，WAEF 仅对文件进行了访问控制，文件安全标签的“域转移”问题只是依靠主体进程和客体文件的安全标签；关于网络的设计，WAEF 对 SOCKET 的控制是在 TDI 层对所有 IRP 包进行拦截，分析和仲裁。WAEF 的所有功能都包含在一个大驱动里面，一些功能也许以后永远也用不到。

3　WEAEF设计改进

针对上一节对 WAEF 的分析，我们设计出了一个改进版本 WEAEF，实现了在兼容性、高效性、稳定性等方面的优化。

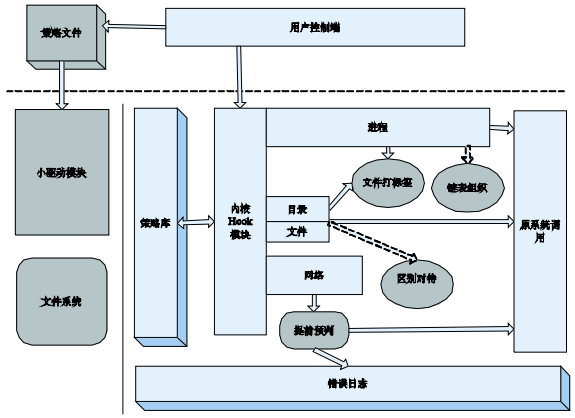

图 3 WEAEF 框架

3.1 驱动的优化

Windows 操作系统允许我们编写自己的驱动程序实现特定的功能，驱动程序编写的一个很大的缺点是不稳定，稍有不慎就会导致所谓的“蓝屏”。越往底层实现，这个问题就变得越加严重。

WAEF 实现了从驱动的加载，运行，卸载等所有功能。文件系统安全标签的初始化是 WAEF 实现访问控制的基础。对于操作系统而言，文件是可以静态存储的，它们的安全标签必然也要静态存储。所以我们要在 WAEF 加载之前就要对文件系统进行初始化，然而文件系统安全标签的初始化工作以后不会再有用，所以把这部分功能放在 WAEF 驱动程序里，不仅仅浪费空间，还降低了稳定性。在 WAEFE 里，我们采用一个小驱动模块专门实现刚开始的文件系统的初始化工作。文件系统的初始化工作完成之后，它会自动卸载，算是完成使命了，然后再加载大驱动模块实现系统的访问控制。

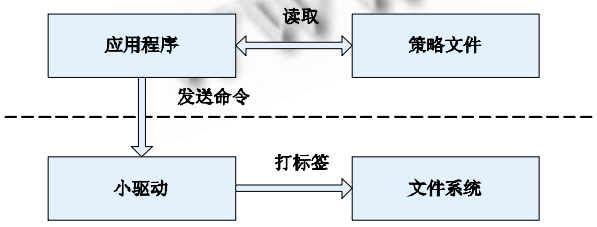

图 4 小驱动模块

3.2 进程的优化

在 Linux 操作系统下，我们为了给进程添加一个数据项，只要修改一下源码，重新编译一下内核就可以实现，然而在 Windows 操作系统中，这是不允许的。Windows 系统中，每一个进程都对应着一个进程描述符，进程描述符在设计之初留下了一些空闲域，我们可以采用这些空闲域来添加进程的安全描述符。但是，这种方法有很大的局限性，因为这些空闲域的相对位移量不是固定不变的。

随着 Windows 操作系统版本的不断更新，进程描述符在不断的变化导致了空闲域的相对位移量也随着变化。为了更好的兼容性，我们这里采取链表式管理进程的安全描述符，让进程描述符与安全描述符间接地联系起来，相互之间不会影响，WAEFE 的兼容性得到了很好的解决。

3.3 文件的扩展

一个安全的操作系统应该尽量多的包含所要保护的对象。对于 Windows 而言，文件和目录是分开的，仅仅针对文件进行访问控制是不全面的，而且目录的安全标签和文件的安全标签非常相关，所以不能分开来看，更不能只对其中一个进行访问控制。

目录对文件的影响主要表现为文件的路径。进程作为主体，文件作为客体，有些访问控制在实施的时候，首先是主体进程对客体目录进行访问控制。比如：在创建文件的时候，这时候文件还不存在，我们进行了创建操作的拦截，进程作为主体，文件所在路径对应的目录是客体，访问策略库进行仲裁判断，允许则在该目录下创建，否则拒绝创建。

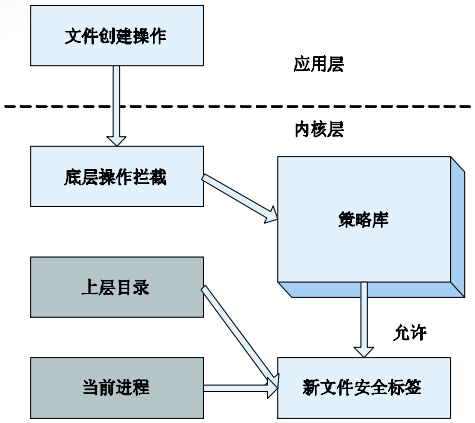

图 5 新建文件安全标签

文件的安全标签不仅决定于创建进程的安全标

签，还受到所在目录的安全标签影响。两者的安全标签共同决定了所创建新文件的安全标签。

3.4 网络的预判

对于网络部分，主要是针对网络套接字 socket 的操作。WAEF 对 socket 的创建、请求、发送、关闭等进行了访问控制，而且对所有的操作都进行了拦截、分析、仲裁。

我们在对客体进行访问控制的时候，最重要的一步是访问策略库进行判断仲裁。而访问策略库是一个非常耗时的操作，有效地减少访问庞大的策略库是提高系统效率的重要方法。经过对 socket 的研究，socket 在创建的时候，仅仅是在本地新建了几个描述和通信的数据结构，并没有真正实施有效地操作。在这个操作拦截中，可以进行拦截、分析，没有必要到策略库当中进行仲裁。

WAEF 只实现了 socket 发送操作的监控，认为发送和接收是成对的，只要监控一个就可以了。其实不然，仅一个操作也可以造成危险的访问。两个操作都要进行监控，不同的操作类型，实施的访问策略是分别对待的。

在进行 socket 的请求，发送，接收等操作的时候，为了实现高效性，我们添加了预判过程，对于网络 socket，其最重要的信息包括：ip 地址，port 端口，type 协议等。在实施访问控制的时候，预判模块首先提取出有效信息做一次过滤，然后再来决定是否有必要进入下一步的提取安全标签访问策略库。这样可以很大程度上的实现高效性。

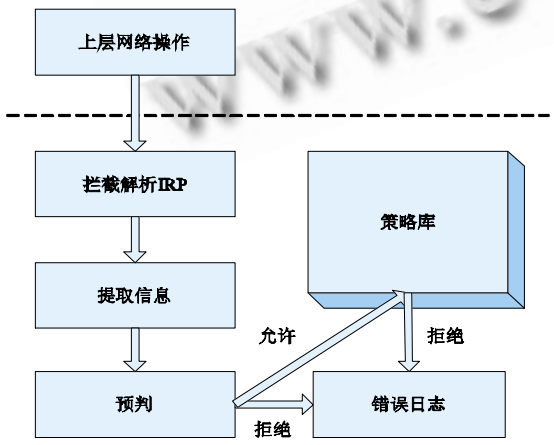

图 6　网络操作预判

4　WEAEF设计实现

4.1 驱动的优化

WEAEF 包括两个驱动模块 WSMSYS 和 MINISYS。WSMSYS 实现了安全模块和策略库的加载以及 Hook 的实施等。MINISYS 实现了文件系统安全标签的初始化。

```
/*MINISYS 提供给文件打标签的接口*/
NTSTATUS SetFileSid(IN HANDLE FileHandle, IN PVOID * sid, ULONG sidLength)
/*调用系统函数给文件打上安全标签*/
status = ZwSetEaFile(FileHandle,&isb, eaBuff, eaLength);
```

MINISYS 实现了文件系统安全标签的初始化工作之后自动卸载，减少了存储空间，降低了原来的大驱动的复杂度，提高了系统的稳定性。

4.2 进程的优化

Windows 操作系统的版本不断地向前发展，描述进程的进程描述符也是不断的变化。我们采取硬编码的方式，仅仅能解决现在的问题，随着系统的变化这种方法就不再适用了。为了更佳的兼容性，我们放弃了硬编码的方法，采用了进程与安全描述符对应的关系表来实现系统无关性。

```
typedef struct dnode
/*定义双向链表结构体*/
{
char context[MAXNAME];
/*进程 Handle 加名字的字符串*/
int SID;
/*进程的 SID*/
struct dnode *prior,*next;
/*定义两指针，分别指向其前驱和后继*/
}dnode;
```

用双向链表来维护进程句柄和进程名字与进程安全属性的关系，进程 Handle 和名字组成的字符串唯一确定了进程的进程描述符。链表式的管理实现系统无关性，很大程度上的改善了兼容性问题。

4.3 文件的扩展

文件和目录在 Windows 操作系统下有很大的区别，虽然目录本身不会存储有效地信息，但是它对文

件的影响是不容忽视的。对安全属性的监视，我们不仅采用了区别对待文件和目录，而且还对它们安全属性之间的关系进行了研究和设计。

我们在底层截获一个文件目录操作的时候，进行区别对待。

```
/*判断是目录还是文件,默认是文件,如果是目录则 fileflag=0。在进行权限检查的时候，文件和目录是不一样的*/
if(fileflag)
    action=pfac_permission_check(sSid,tSid, SECCLASS_FILE, FILE__CREATE);  //file
else
action=pfac_permission_check(sSid,tSid, SECCLASS_DIR, DIR__CREATE);  //dir
```

对文件和目录打上不一样的标签，区别对待是为了实现文件的域转移问题。作为主体进程对客体文件的操作权限，不仅仅取决于用户的权限和进程的权限，还受到所在目录的权限的约束，即当前进程对所在目录的权限。

```
/*读取父目录安全标签*/
    Status = QueryFileSid(zFileHandle, tContext, &tContextLength);
```

在当前进程对文件所在目录的权限检查通过之后，如果是创建文件，这里就存在一个给新建文件打上安全标签的问题。

```
/*域转移成新创建的文件的标签*/
    if(fileflag)
pfac_transition_sid(sSid, fSid, SECCLASS_FILE, &tSid);
```

由当前进程和父目录的安全属性产生出一个新的安全属性，给新创建的文件打上一个新的安全标签。

```
SetFileSid(*FileHandle,tContext,tContextLength);
```

4.4 网络的预判

针对网络 socket 不同的操作我们采取了不同的设计方案，主要是为了更佳的安全性和高效性。

Socket 在创建的时候没有做实质性的连接操作，只采取截获并解析。Socket 的请求，发送，接收等操作需要作进一步的处理。解析出来 socket 的相关信息，比如：地址、端口和协议。首先根据这些信息进行过滤，如果被拒绝则返回，否则进一步访问策略库，利用主体进程与客体 socket 的安全标签对进行仲裁。

```
/*提出取 socket 信息*/
ip = (PTDI_ADDRESS_IP) &pAddress->Address;
/*socket 地址，端口信息*/
net_list_add(ip->in_addr,ip->sin_port,0,currentProcessId,socketsid);
```

如果被允许，则进一步访问策略库进行二次审核。

```
/*访问策略库，进行仲裁*/
pfac_permission_check(sid,socketsid, SECCLASS_SOCKET,SOCKET__CONNECT);
```

访问策略库是一个很耗时的过程，所以我们要尽量减少该操作。在实施访问之前，我们进行了首次过滤，减少了访问策略库的次数，提高了系统的性能。

5 性能比较

在稳定性方面，我们反复对该系统进行了测试，未发生过一次蓝屏。这对于 Windows 操作系统而言意义重大，庞大驱动程序功能的缩减，巧妙的设计提高了系统稳定性。由于采用了进程安全描述符存储系统无关性的方法，系统兼容性有了很大的提高，在多个版本的 Windows 操作系统都可以安装实施。

表 1 WAEF 和 WEAEF 兼容性比较

	2000	2003	Xp
WAEF	N	N	Y
WEAEF	Y	Y	Y

在性能方面，我们针对网络部分进行了测试。在上层激发多个网络 socket 通信之前，我们在该系统中要对网络的预判进行配置，该配置文件相对于策略文件非常小。在系统层次，相对于一个存放几十万条策略的文件，访问一个有几百条策略的文件所花费的时间是可以忽略不计的。所以在这里，访问次数可以表示性能的高低。

表 2　WAEF 和 WEAEF 访问策略库次数比较

访问策略库次数	Socket()	其他	合计
WAEF	1000	50000	51000
WEAEF	0	47651	47651
性能提升	100%	4.70%	6.57%

socket 在创建的时候，我们完全去除了该操作访问策略库的可能性。同时在对其他网络操作 connect、bind、send、receive、close 等都进行了预判，大大的减少了访问策略库的次数，从而提高了系统的性能。

6　总结

本文通过对强制访问控制机制和通用的访问控制实施框架 WAEF 的研究，提出了一个基于 Windows 操作系统的改进版设计框架。该设计框架在对 Windows 系统的进程，文件，网络 socket 以及安全模块功能划分进行了改进设计，提高了系统的稳定性，高效性。同时设计实现了 MINISYS 模块，分担了 WSMSYS 的功能，节省了存储空间，降低了冗余，提高了稳定性。下一阶段的研究主要包括：对系统的安全对象的操作实现二次扩展，并可以自由灵活配置安全度；继续完备该系统的日志审计功能；对网络部分的 socket 的操作和安全项进行扩展。

参考文献

1 刘邦明,邬浙艳,孙黄杰.SSDT 挂钩：基于 Windows 内核的 RootKit 技术样本.网络安全技术与应用, 2009,(3):62－64.

2 Li L, Wu J, Guo XW, Li M, Niu XM. Framework for Windows Password Function Security Enhancement. Journal of Southeast Univesity, 2007,37(z1):26－28.

3 Peter L, Stephen S. Integrating Flexible Support for Security Policies into the Linux Operating System. the FREENIX Track. 2001 USENIX Annual Technical Conference. Berkeley. CA: USENIX Association, June 2001:29－42.

4 Zhai GS, Li YD. Analysis and Study of Security Mechanisms inside Linux Kernel. Security Technology, 2008. SECTECH '08. Washington DC: IEEE Computer Society, 2008:58－61.

5 李奇,周学海,陈香兰.Windows 访问控制实施框架的研究与设计.计算机系统应用, 2009,18(12):83－87.