Windows 访问控制实施框架的研究与设计①

Study and Implementation of Windows Access-Control Enforcement Facility

李 奇 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: 在目前的研究中，多采用访问控制安全模型来增强系统安全。Linux 下主要是基于 LSM 实现已有的众多安全模型。但 Windows 下的各安全项目底层实现方式各不相同，缺乏通用的访问控制实施框架。针对 Windows 下安全项目的需要，提出了一套通用的 Windows 实施访问控制策略的框架。它通过修改内核数据结构来存储安全标签，能够提供一系列安全 Hook 供安全模块使用，可以用作安全模块研究与开发的平台。

关键词: 访问控制框架 Hook Windows

1 引言

操作系统的安全特性从操作系统诞生起就成为研究人员关注的焦点。通过访问控制对系统进行加固[1]，能够有效地增强系统安全性。BLP、BIBA、DTE 等安全模型的提出，标志着对于访问控制的研究已经日趋成熟。

Linux 系统下已经实现了众多的安全模型以进行系统加固，如 Selinux、DTE、Smack、Apparmor 等。Linux 内核 2.6.X 版本提供了 LSM 框架[2,3]，使各个安全模块能以内核模块的方式加载，大大提高了 Linux 访问控制机制的灵活性和易用性。为支持各种安全模型，LSM 提供了一系列 Hook 接口，以虚函数的形式保存在结构体 Security_ops 中[2]。基于 LSM 实现的安全模块需要实现这些虚函数。用户进程执行请求执行系统调用过后，会首先定位和查找要访问的 inode，并进行错误检查，之后还需要通过 Linux 自主访问控制的进一步检查。然后，LSM 的 Hook 函数会在内核执行实际操作之前，将操作截获，并提供该操作主客体的安全标签，操作类型等内容，将其提供给安全模块。最后由安全模块匹配其中的策略决定是否允许执行，决定是否允许该操作执行。FreeBSD 也提供了 TrustedBSD MAC 框架，它与 LSM 实现同样功能[4,5]。

与 Linux 相比，Windows 系统不支持强制访问控制，并缺少支持不同访问控制模型的通用框架，导致在 Windows 上实施安全模块时，需要自行设计底层实现，重复了大量工作。本文通过研究 LSM 的设计思想、Windows Hook 技术以及访问控制机制的实现，提出了一个通用的访问控制实施框架(Windows Access-control Enforcement Facility Framework，WAEF)。WAEF 可以加载各种安全模块，并为这些安全模块提供底层支持。正文第 2 节介绍 WAEF 框架的设计；第 3 节介绍了 WAEF 安全标签的设置；第 4 节详细描述了 WAEF 的实现细节，包括模块的启动，策略的加载，以及 Hook 点的选取；第 5 节给出了性能评价；最后是总结与展望。

2 WAEF框架的设计

WAEF 是基于系统调用的访问控制实施框架，它运行于内核态，为安全模块提供对内核对象的访问接口。用户进程请求的系统调用在内核态被 WAEF 的 Hook 函数截获，由安全模块来决定是否允许操作。WAEF 的 Hook 架构如图 1 所示。

在安全操作系统的运行过程中，需要根据不同的安全需求加载不同的安全模型，因此，WAEF 框架需

① 基金项目:电子信息发展基金(财建[2008]329,工信部运[2008]97)
收稿时间:2009-04-03

© 中国科学院软件研究所 http://www.c-s-a.org.cn

要提供安全模块组合机制。LSM 采用栈式方法[2]，通过由主安全模块向从安全模块提供接口的形式实现多安全模块的加载。当加载多个安全模块时，需要进行大量的接口编写工作。

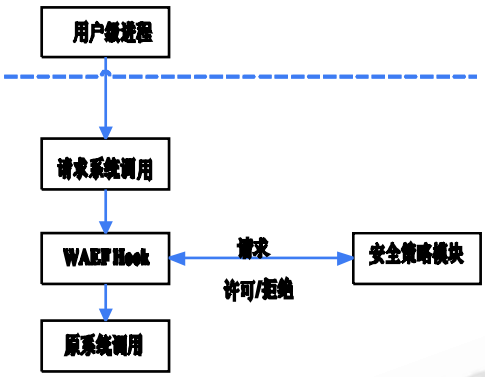


图 1　WAEF Hook 架构

WAEF 采用链式方法实现多安全模块加载，如图 2 所示。各安全模块地位平等，决策的优先级由核心安全服务器(Core Security Server, CSS)决定，与加载的顺序无关。在加载多安全模块的情况下，CSS 需要实现仲裁功能，获得各安全模块的决策结果并做出最终判断。

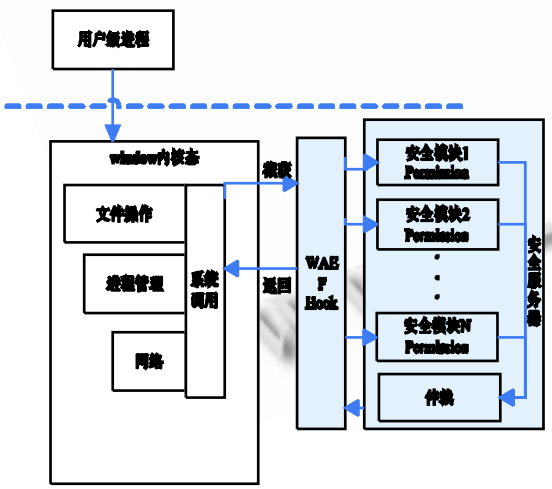


图 2　WAEF 多安全模块支持

WAEF 用 Security_ops_entry 和 Security_context_entry 支持链式加载安全模块的方法，如图 3 所示。CSS需要实现 Security_ops_entry 中所有的 Hook 接口，CSS 实现的各 Hook 函数获取各安全模块的对应函数的判定结果，并做出最终决策。Security_context_entry 存储对应安全模块的安全标签。

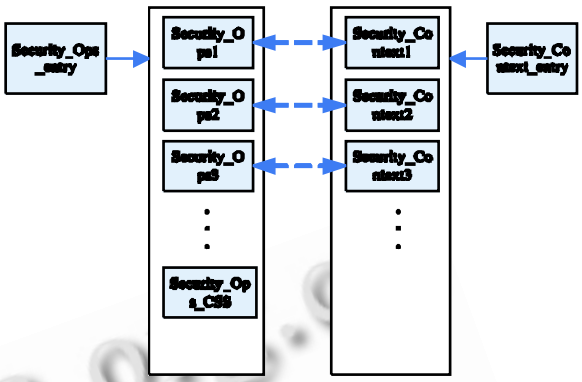


图 3　Security_ops_entry 与 Security_ context_ entry

采用链式方法加载安全模块的好处在于，只需要安全服务器实现对多模块的接口，不需要由一个安全模块为另一个安全模块提供接口，简化了接口实现。栈式加载安全模块的方法需要修改主安全模块标签结构，以存储从安全模块标签。链式加载则可以自动管理安全标签与安全模块的对应关系。

3　WAEF安全标签设置

3.1 进程标签

Windows 操作系统中，结构体_EPROCESS 描述了进程。为了维护进程安全信息，WAEF 提供函数使用_EPROCESS 的成员 Peb 的 systemReserved 保存进程的安全标签，在进程创建时通过 SetTagProcess 函数进行设置，函数定义为：

$$NTSTATUS\ SetTagProcess(IN\ PHANDLE\ processHandle,\ IN\ PVOID\ tag,\ IN\ int\ amId);$$

进程结束时，通过 TemiTagProcess 函数通知安全模块，定义为：

$$NTSTATUS\ TemiTagProcess(IN\ PHANDLE\ processHandle,\ IN\ int\ amId);$$

其中，参数 amId 指明标签属于第几个安全模块。

3.2 IPC 标签

在 Windows NT4.0 以上的操作系统中，进程间通常采用端口通信、共享内存、管道技术、邮件通道、

© 中国科学院软件研究所 http://www.c-s-a.org.cn

网络接口、映像文件等方式进行通信。网络接口通信属于网络 Hook 部分，管道技术、邮件通道属于文件 Hook 部分。其余几种通信方式涉及到的内核对象有 Port、Semaphore、Section、Event。出于性能考虑，WAEF 提供的 IPC-Hook 将涉及的对象名和对象 Handle 值传给安全模块，由安全模块自行维护安全标签。

3.3 文件标签

WAEF 现阶段只实现了对 NTFS 文件系统的支持。可以通过实现多文件系统安全标签设置与获取函数来支持多文件系统。

NTFS 文件系统中，文件的一切都由属性描述，扩展属性不是固定的，可以随时增加。当文件或文件夹小于 1500 字节，所有属性都会常驻主文件表(MASTER FILE TABLE， MFT)中。即使文件属性不能保存在一个 MFT 记录中时，也可以使用 AttributeList 对其进行管理。因此可以方便地把文件安全标签作为文件的扩展属性加入 MFT 中。

文件的安全标签在安全模块加载时或文件创建时使用函数 SetTagFile 进行设置，函数定义为：

```
NTSTATUS SetTagFile(IN PHANDLE fileHandle,
                    IN PVOID tag,
                    IN int amId);
```

安全模块卸载的时候需要使用函数 RemoveTagFile 卸载对应的安全标签，函数定义为：

```
NTSTATUS RemoveTagFile(IN HANDLE fileHandle,
                       IN int amId);
```

3.4 注册表标签

注册表是 Windows 操作系统中存放各种参数的核心数据库，直接控制 Windows 启动、驱动的装载、应用程序的运行，在系统规划中起着核心的作用。

由于在注册表中，各键值的路径和名称是唯一的，因此，WAEF 提供的注册表操作 Hook 将涉及的注册表项路径和名称传给安全模块，由安全模块自行维护安全标签。

3.5 网络标签

通常安全策略需要协议、端口、访问网络的进程和访问目标机或目标进程的信息作为决策分析的因素。WAEF 在 TDI 层使用虚拟设备截获 IRP，从 Windows 提供的保存应用程序与打开的地址及端口信息的关联关系链表 address_entry 中获取信息并传递给安全模块。安全标签由安全模块自行维护。

4 具体实现

WAEF 是实验和开发各种安全模型的一个平台，所以它必须尽早启动，能够支持多安全模块加载，选择合适的 Hook 方法来提供 Hook 接口，Hook 接口集要尽量完整[6]。本节介绍 WAEF 的启动，安全模块的加载与卸载，Hook 方式的选择以及 Hook 点的选取。

4.1 WAEF 启动

基于 WAEF 的安全模块以服务的形式运行于系统内核态，需要用户拥有 Adiministrator 权限，以进行服务的安装。安全模块越早启动，越能给系统更好地保护。步骤如下：

(1) 使用服务控制管理器(Service Control Manager，SCM)来安装 WAEF_Loader 服务。

(2) 使用 NativeAPP 编写服务启动程序并在注册表中 BootExec 项注册，以便在系统登陆前启动 WAEF_Loader 服务。

在启动 WAEF_Loader 服务时，必须加载 CSS 提供的 Security_ops，否则系统无法正常运行。然后再按需要加载安全模块。注意，加载或卸载安全模块后，系统需要重启以更新 Security_context_entry。

4.2 安全模块加载与卸载

WAEF 加载与卸载分为安全服务器(CSS)和安全模块两类。

CSS 的加载过程即使用函数为 Security_ops_entry 中 Security_ops_CSS 项赋值的过程。通常在 WAEF 安装时或 CSS 在加载新的安全模块后更新时调用。

```
NTSTATUS Register_css(IN Security_ops*);
```

安全模块的加载过程即向 Security_ops_entry 链表添加安全描述结构并在下次启动时为系统主客体的 Securty_context_entry 增加安全标签的过程，卸载即从 Security_ops_entry 链表删除安全描述结构

© 中国科学院软件研究所 http://www.c-s-a.org.cn

并移除系统主客体的 Security_context_entry 中对应的安全标签的过程。WAEF 提供了两个函数加载或卸载安全模块：

NTSTATUS Register_security(IN Security_ops);*
NTSTATUS unRegister_security(IN Security_ops);*

WAEF的Hook函数获取当前系统调用中主客体安全标签并由当前安全模块判断操作，将结果提交给安全服务器，在所有策略模块判断结束后，根据安全服务器的决策结果返回或运行系统调用。如图 4 所示：

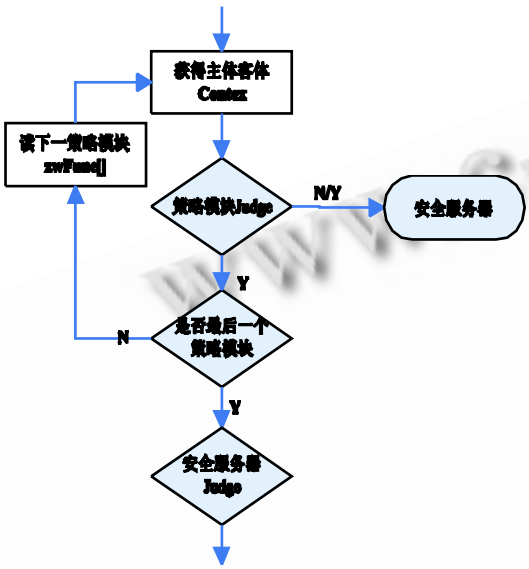


图 4　WAEF Hook 多安全模块工作流程

4.3 WAEF_Hook 设计

4.3.1 WAEF_Hook 方法

Windows 下常见的 Hook 方法都是通过在 Windows 执行系统调用的不同阶段对操作进行截获来达到目的，如图 5 所示：

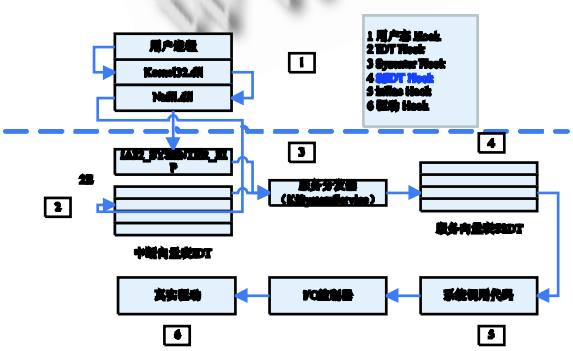


图 5　常用 Hook 方法

其中用户态 Hook 处于 ring3 级，无法有效地保护系统。驱动 Hook 方式容易引起系统不稳定。Sysenter-Hook 和 IDT-Hook 只能控制执行流，无法对返回结果进行修改。Inline-Hook 对操作系统版本依赖较大，存在一定的通用性问题。

SSDT-Hook 处于内核态，并且在各版本 Windows 中实现方法基本相同。WAEF 只需额外维护 SSDT 表与 security_ops_entry 的对应关系。因此，WAEF 选择使用 SSDT-Hook 方式。

TDI-Hook 和 NDIS-Hook[7] 相结合的方法对 Windows 网络操作进行监控的技术已经成熟，WAEF 也采用这种方法实现网络的 Hook 接口。

4.3.2 进程/线程操作 Hook

WAEF 提供的进程 Hook 函数覆盖了进程的整个生存周期。创建进程时，截获系统调用 ZwCreateProcess_Hook 或 ZwCreateProcessExHook，由安全模块判断是否允许操作。如果允许，安全模块使用 WAEF 提供函数在_EPROCESS 成员 peb 的 systemReserved 中添加新进程的安全标签。进程创建成功后，打开(ZwOpenProcess)、(ZwSuspendProcess)、唤醒 (ZwResumeProcess)、结束(ZwTerminateProcess)对应的 Hook 在系统调用执行前执行，将进程安全标签提交安全模块进行决策。WAEF 提供的线程 Hook 函数与进程 Hook 函数类似，同样覆盖了创建、打开、挂起、唤醒、结束。

4.3.3 IPC Hook

WAEF 为安全模块提供了 Port、Section、Semaphore、Event 的创建、打开等系统调用 Hook。通过这些 Hook 函数，安全模块可以控制进程对这些内核对象的使用。

当创建一个 IPC 对象时，WAEF 向安全模块提供当前进程的安全标签，由安全模块进行判断。如果允许创建，WAEF 将创建的安全对象的名称以及 Handle 值提供给安全模块，由安全模块自行维护安全标签。安全模块需要根据其维护的安全标签以及安全模块本身的策略库进行决策，将结果返回给相应的 Hook 函数。

4.3.4 文件操作 Hook

WAEF 提供的文件操作 Hook，包括文件的创建

© 中国科学院软件研究所 http://www.c-s-a.org.cn

ZwCreateFile)，打开(ZwOpenFile)，读写(ZwReadFile,ZwWriteFile,ZwQueryAttributesFile,ZwQueryDirectoryFile,ZwQueryFullAttributesFile,ZwSetInformationFile)等操作。

WAEF 还提供了相应的 Hook 使安全模块可以对进程使用邮件通道(ZwCreateMailslotFile)以及管道技术(ZwCreateNamedPipeFile)进行通信的行为进行控制。

4.3.5 注册表操作 Hook

注册表是 Windows 操作系统中存放各种参数的核心数据库，在系统运行时，涉及注册表的系统调用较为频繁。WAEF 提供了所有的注册表操作 Hook 供安全模块使用，包括注册表的创建、打开、修改、重命名、查询、刷新、删除、导出、导入等，共计 23 个。其中部分系统调用存在直接的因果关系，例如：ZwOpenKey 规定了访问权限，其他对注册表的操作必须遵循这个访问权限。安全模块可以根据需要选择部分接口实现。

4.3.6 网络 Hook

WAEF 在 TDI 层通过截获 IRP 请求，获得应用与下层协议栈之间交互的命令和数据，并向安全模块发出决策请求。拦截的 IRP 请求包括创建(IRP_MJ_CREATE)、关闭(IRP_MJ_CLOSE)、关联请求(TDI_ASSOCIATE_ADDRESS,TDI_DISASSOCIATE_ADDRESS)以及 Socket 操作。

WAEF 通过解析 address_entry 可以得知当前操作的目的是 IP 或者本地进程。安全模块可以对进程使用网络接口进行通信的方式进行控制。

4.3.7 其他 Hook

除了 4.3.2 -4.3.6 中介绍的系统调用 Hook，WAEF 还提供了 SSDT 表中剩余的绝大部分 Hook 函数。包括启动项操作 Hook、调试操作 Hook、内存操作 Hook、权限检查操作 Hook、服务及设备操作 Hook 等以及涉及 4.3.2 -4.3.6 方面不太重要或功能重复的共计 276 个 Hook 函数。

5 性能分析

在 VMWare 中，采用第 4.1 中建议的方式启动 WAEF 服务，加载全域对全类型允许的简单安全模块并实现所有的 Hook 函数接口，通过 RDTSC 指令计算每类系统调用消耗的平均 CPU 周期。将 CPU 周期分为 3 个部分：原系统调用部分、WAEF 取得安全标签部分和安全模块决策部分。取原始系统调用 CPU 周期数与 WAEF 取得安全标签周期数进行对比，结果如所示：

表 1 原系统调用与 WAEF 使用系统周期对比

类型	原系统调用	WAEF 取得安全标签	负载(百分比)
进程	135119	20827	15.4%
IPC	88344	10322	11.7%
文件	233526	55538	23.8%
注册表	78798	9226	11.7%
网络	-	15252	
其他	133592	27496	20.6%

可以看出，如果对所有的系统调用进行 Hook 将对系统造成较大的影响。

由于从注册表项 Handle 获得键路径和名称操作较快，因此 WAEF 耗费的 CPU 时间相对较少。进程系统调用和 IPC 系统调用只占系统运行时系统调用总数较少一部分，因此不需要更多优化。网络部分使用的是 TDI-Hook，只截取了 IRP 包，无法获得网络操作原来所需的时间，但 WAEF 获取安全标签所用的 CPU 周期较少。

获取文件操作所涉及到主客体的安全标签需要两个步骤：获取进程的_EPROCESS，再从_EPROCESS 中读取标签；通过参数 Handle 找到文件，读取文件扩展属性中的安全标签。其中读取文件扩展属性占用系统调用周期较多。

所幸的是，许多频繁发生的文件操作如 ZwWriteFile、ZwReadFile、ZwWriteFileGather、ZwReadFileScatter 等操作可以通过 Hook 其他系统调用来达到对其进行阻止的目的。例如，安全模块通过对 ZwOpenFile 进行 Hook，可以在文件打开时就检查的进程读写的权利，根据其打开方式决策是否允许操作，不需要对上述频繁发生的读写系统调用进行 Hook。

(下转第 102 页)

© 中国科学院软件研究所 http://www.c-s-a.org.cn

(上接第 87 页)

操作系统安全加固软件研发及产业化项目(电子信息产业发展基金)windows 下强制访问控制模块的执行部分基于 WAEF 实现，仅使用了 WAEF 提供的 276 个 Hook 函数中的 61 个，在不加载安全模块的情况下对系统负载仅增加 4%。

6 总结与展望

本文提出了一套通用的运行于内核态的 Windows 安全模块框架WAEF，该框架提供了一系列安全 Hook，能够支持目前大多数安全模型。使用WAEF 可以方便地进行强制访问控制、自主访问控制的开发和研究，已应用于电子信息产业发展基金(操作系统安全加固软件研发及产业化)项目中。其他的安全机制如审计系统、主机入侵检测等开发和研究也可以使用 WAEF。

参考文献

1 胡东辉,周学海.计算机安全模型研究.小型微型计算机系统, 2005,26:561－567.

2 Wright C, et al. Linux security modules: general security support for the linux kernel. Foundations of Intrusion Tolerant Systems, 2003 [Organically Assured and Survivable Information Systems]. 2003.213－226.

3 Spencer RSS, Loscocco P, Hibler M, Andersen D, Lepreau J. The flask security architecture: System support for diverse security policies. 8th USENIX Security Symposium. Washington,D.C.,USA:Usenix: 123－139.

4 Watson R, et al. Design and implementation of the trusted BSD MAC framework. DARPA Information Survivability Conference and Exposition. 2003.38－49.

5 Watson R, et al. The TrustedBSD MAC framework. DARPA Information Survivability Conference and Exposition. 2003.13－15.

6 Kuperman BA. A categorization of computer security monitoring systems and the impact on the design of audit sources. 2004, Purdue University.151.

7 刘鹏远. Windows 平台通用个人防火墙的分析与实现.计算机系统应用, 2008,17(8):109－111,127.

© 中国科学院软件研究所 http://www.c-s-a.org.cn