第36卷 第14期 计 算 机 工 程 2010年7月
Vol.36 No.14 Computer Engineering July 2010
·安全技术· 文章编号：1000—3428(2010)14—0127—02 文献标识码：A 中图分类号：TP309

轻量级虚拟机系统资源保护层研究

申文迪，罗克露
(电子科技大学计算机学院，成都 610000)

摘 要：为了保护计算机不受未知恶意软件的破坏，采用模拟系统环境虚拟执行保护计算机关键资源，通过 HOOK 技术引入轻量级资源虚拟机，实现对计算机关键资源的保护。提供多种行为分析 API，使之成为一个可供二次开发的分析平台，成功地解决了无特征码情况下新型木马的识别问题。

关键词：虚拟机；资源保护；计算机安全

Research of System Resource Protection Layer on Lightweight Virtual Machine

SHEN Wen-di, LUO Ke-lu
(College of Computer, University of Electronic Science and Technology, Chengdu 610000)

【Abstract】 In order to protect the computer from the destruction of the unknown malicious software, this paper carries out the idea of using virtual executing in a simulated system environment to protect computer resources. A lightweight virtual machine is introduced with HOOK to protect critical resources, and a series of API interfaces are provided to make this lightweight virtual machine an open platform available for secondary development. The method successfully identifies new Trojan without feature code.

【Key words】 virtual machine; resource protection; computer security

1 概述

随着信息化技术的深入和互联网的迅速发展，计算机网络已经成为国家的经济基础和命脉。不过，也为越来越多的病毒、木马的传播创造了途径。恶意软件也是信息时代的产物，它充分利用 Internet 带来的方便进行自己的恶意活动。恶意软件的另外一个严重后果是可以利用系统漏洞提升自己的运行权限，然后从用户电脑中窃取自己想要的隐私信息，如密码、银行账号。

以上只是众多恶意代码中的很小一部分。为了保护公民自身的利益，必须对软件中可能出现的恶意行为进行有效的防范，因此，本文采用资源虚拟化的操作系统抽象层虚拟机防范恶意软件的破坏。

2 操作系统资源虚拟化保护基本原理

在计算机领域经常用到有关虚拟机的概念，如 Java 虚拟机、Windows9X 的虚拟及一些虚拟设备。虚拟机是相对于物理存在的计算机而言的，通常所说的虚拟机，如 VMWare，是指虚拟出一个 CPU 运行整个操作系统，包括软件、硬件等各方面的虚拟化。它主要是指运行在 Windows 或 Linux 计算机上的一个应用程序，这个应用程序“模拟”了一个基于 x86 的标准 PC 环境[1]。

操作系统资源虚拟化也称为虚拟机技术，但又不同于上述虚拟机，而是指在同一操作系统上的一种通过应用命名空间虚拟化(Namespace Virtualization)技术模拟的虚拟环境，从而达到资源保护与隔离作用，就像在另一台机器上运行程序一样，程序对系统等的操作不会影响到本机。这样，就可以为执行的程序提供一个安全执行环境。这个系统称为 OSAL (OS Abstract Layer)[2-3]。

命名空间虚拟机并不是最新的，在 Solaris Containers, Linux VServer 和 FreeBSD Jails 系统中，已经采用了这种技术。但是，作为全世界应用最广泛的系统，Windows 仍有一些技术难题未解决而没有引入此技术。原因有以下 3 点：

(1)Windows 系统有太多类型的命名空间。最简单、最常用、最重要的命名空间是文件和注册表资源，但仅仅虚拟这 2 个命名空间是远远不够的，达不到想要的隔离效果。

(2)Windows 系统中的特殊管理机制。Windows 系统有一系列特殊管理机制的守护服务，其中一些服务与内核同等重要，很难复制到每个虚拟机中，所以，虚拟命名空间必须处理这些由共享进程创建的特殊资源。

(3)Windows 系统的进程间通信机制。Windows 系统存在大量特殊的进程间通信机制，而其中某些不直接依赖于名字，必须特殊处理这些机制以达到虚拟机的完美隔离。

3 资源虚拟化具体实现

3.1 文件系统虚拟化

文件系统虚拟化是虚拟机中最重要的部分，它的主要作用是将系统中的一些待保护区域的文件、目录及一些特殊的设备文件如命名管道、邮槽与真正的主机系统隔离。要达到隔离的目的，可以通过 HOOK 技术截获系统调用并改写其命名空间。

对于普通文件(文件、目录)，命名空间虚拟化比较容易实现，只要改变传入的文件路径即可，不过，效率方面的问

基金项目：国家“242”信息安全计划基金资助项目((242)2007B30)
作者简介：申文迪(1985－)，男，硕士，主研方向：嵌入式操作系统，网络安全；罗克露，教授、博士
收稿日期：2009-12-23 **E-mail：**swdkk@qq.com

题也出现了，如果文件比较大，复制文件是一种不能容忍的高开销，为了减少这种情况，采用了写时拷贝机制(Copy on Write)，这种写时拷贝机制不同于内存管理的写时拷贝机制，因为 Windows NT 基于块方式的文件写时拷贝机制需要处理底层文件系统相关的各种数据结构以及很多复杂的机制，所以为了简化逻辑和保持文件的完整性，采用了单个对象为单位的写时拷贝机制。当进程以写入的方式打开一个文件时，会截获其传入的文件路径，并且按照自己的命名规则将其拷贝到 OSAL 的目录下，如 C:\OSAL，并且，该文件的属性及其所有的父目录都将被复制，例如，主机上的 C:\Windows\SYSTEM32\cmd.exe 文件以写入方式被打开，将其拷贝到目录中，路径为：C:\OSAL\HardVolume1\Windows\ SYSTEM32\cmd.exe。复制完成后，将恢复被截获的函数执行，当然，在执行之前，将其传入路径用“虚拟”路径代替。当进程创建一个新的文件时，处理过程与写入相同。

对于特殊的设备文件，如命名管道和邮槽的虚拟化，本文的处理方式与普通文件类似，也是通过在虚拟目录下创建对应的文件，如 OSAL 中进程调用创建\\.\pipe\mypipe01，在虚拟目录下创建对应的文件，如 C:\VMStore\Device\Pipe\mypipe01。如果此虚拟机中另一个进程请求打开该命名管道，OSAL 会将此请求重定向为打开上述路径文件，这样，通过使用文件虚拟命名管道，对管道的操作实际上就是对文件的操作，当管道被关闭，该目录及下面的文件也会被完全删除。

表 1 给出在实现文件系统虚拟中 HOOK 的几个系统底层 API。

表 1　文件系统虚拟化关键函数

函数名	作用
ZwCreateFile	创建文件
ZwOpenFile	打开文件
ZwQueryInformationFile	通过一个文件的句柄获得对应的各种信息
ZwQuerydirectoryFile	通过一个目录句柄返回在这个目录下的所有文件
NtsetInformationFile	可能被用来删除或者重命名一个文件/目录

3.2　注册表虚拟化

注册表是 Windows 操作系统中极其重要的一个部分，它几乎保存了所有与操作系统及应用程序等相关的配置和信息。通过注册表可以修改很多操作系统属性，或者将自己的应用程序赋予一定的权限，所以，安全环境必须保护注册表。本文同样采用虚拟化实现 OSAL 中程序所产生的注册表操作与主机系统中原有的注册表信息隔离。

Windows 注册表系统虽然底层存储方式为一种特殊文件，与平时接触的文件系统不一样，但是操作注册表又与普通文件系统十分类似，例如注册表中的项都有自己的路径，项在打开或创建操作中都是通过返回句柄标识的。所以，在 OSAL 实现中，实现方式与文件系统极其类似，也为注册表虚拟了一个根键，如“\HKEY_CURRENT_USER\OSAL”。所有新创建及在写时拷贝而被复制的注册表键都会放在这个根下面，并且其完整路径将被复制。由于实现和文件虚拟化极其类似，因此不再赘述具体虚拟化过程，可以参考前一节的实现。

整个注册表虚拟化的实现使用了系统调用截获技术，在 Windows 的系统调用层截获并控制了几个相关的注册表操作系统调用。当 OSAL 截获这些调用对应的请求时，第 1 步检测是否为 OSAL 中进程发起的调用，如果不是，则不进行任何操作，直接恢复函数执行，若是，则根据具体指令的不同进行不同的处理。由于注册表虚拟和文件虚拟的强类似性，因此注册表操作和文件操作处理方式一一对应，见表 2。其具体实现类似于上一节的描述。

表 2　注册表与文件操作系统调用对应关系

注册表操作	文件/目录操作
ZwCreatKey	ZwCreatFile
ZwOpenKey	ZwOpenFile
ZwDeleteKey	ZwSetInformationFile-删除请求
ZwDeleteValueKey	ZwSetInformationFile-删除请求
ZwSetValueKey	ZwSetInformationFile-重命名请求
ZwEnumerateKey	ZwQueryDirectoryFile
ZwEnumerateValueKey	ZwQueryDirectoryFile
ZwQueryKey	ZwQueryInformationFile
ZwQueryValueKey	ZwQueryInformationFile

在注册表中，可以将注册表键看作是文件系统中的目录，注册表项看作是文件系统中的文件，注册表值看作是文件系统中文件的内容。

3.3　内核对象虚拟化

在 Windows 中，内核对象是以目录的方式层次管理的，所有内核对象都有一个共同的根目录，以“\”表示。每个内核对象都有自己的子目录，例如一个事件对象可以放在“\BaseNamedObjects\”目录下。一般来说每种内核对象都有其固定的目录，如事件对象、互斥对象、信号量对象、区域对象等都会放在“\BaseNamedObjects\”目录下，而端口对象根据其用途不同被放在“\Windows\”和“\RPC Control\”目录下。这样内核对象的命名空间虚拟化方法与文件系统的方法完全一样，通过重定向内核对象的根目录达到虚拟化的目的。例如有一个命名事件对象 Event1，在其被创建时 Windows 会默认将路径设置为“\BaseNamedObjects\Event1”，而被 AVM 在系统调用层重定向后，其路径就为“\OSAL\BaseNamedObjects\Event1”。这样该事件对象仅在 AVM 中可以被 OpenEvent 等 API 函数打开，而对于主机系统，它对用户态的应用程序是透明和不可访问的。

整个内核对象虚拟化的实现使用了系统调用截获技术，在 Windows 的系统调用层截获并控制了几个相关的内核对象创建和打开操作系统调用。当 OSAL 截获这些系统调用对应的请求时，首先检查发起请求的进程是否为被虚拟化的进程。如果不是，则该系统调用被直接传递到 Windows 的原始系统调用函数；否则，根据不同的请求进行不同的处理。表 3 列出了实现内核对象虚拟化需要截获的函数。

表 3　内核对象的创建操作

对象类型	创建操作
互斥对象	ZwCreatMutant
信号量对象	ZwCreatsemaphore
定时器对象	ZwCreattimer
区域对象	ZwCreatSection
端口对象	ZwCreatPort

3.4　分析接口的提供

因为在实现操作系统资源虚拟化的同时会大量 HOOK 操作系统底层函数，这样，应用程序对操作系统底层的操作也完全在 OSAL 的掌控之中，所以可以提供 API 返回当前应用程序对底层进行的操作，这样，OSAL 就不仅仅是一个资源保护工具，而且是一个资源保护平台，可以提供 API 供二次开发，甚至，通过编写规则，还可以基于 OSAL 实现 HIPS 系统。而这些 API 实现也很简单，就是在进行资源虚拟之后，在恢复函数正常执行之前，通过统一的格式，生成一个特定的结构，存入内存中，供查询 API 取出应用程序操作。

(下转第 131 页)