一种基于交叉视图的 Windows Rootkit 检测方法

白光冬 郭 耀 陈向群
(北京大学信息科学技术学院软件研究所高可信软件技术教育部重点实验室 北京 100871)

摘 要 Rootkit被病毒、木马等恶意软件用来隐藏其在被入侵系统上的踪迹,使得它们能够在系统中潜伏较长时间,它的存在给系统及其使用者带来较大的安全隐患。首先对Windows rootkit进行了研究,以此为基础,从rootkit的行为入手,提出了基于进程检测进行rootkit检测的机制,并设计了一种基于交叉视图的Windows rootkit检测方法。这种方法通过比较从系统高层和底层获得的进程列表,从中检测出被rootkit隐藏的进程,其中,系统底层的进程列表通过在Windows内核中查找内核对象的方法获得。最后,利用这种方法实现了一个Windows rootkit检测工具VITAL,并选用若干有代表性的rootkit进行实验,通过和其他工具的对比,验证了这种方法具有较强的检测功能。

关键词 rootkit,rootkit检测,隐藏进程
中图法分类号 TP309.5 **文献标识码** A

Windows Rootkit Detection Method Based on Cross-view

BAI Guang-dong GUO Yao CHEN Xiang-qun
(Key Laboratory of High Confidence Software Technologies, Ministry of Education, China Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract Rootkits are often used by attackers to hide their trails in an infected system, so attackers can lurk in an invaded system for a longer time. Rootkits have become new threats to the security of OS. This paper first presented a summary on rootkit and rootkit detection technologies on Windows. Built on the existing techniques, this paper proposed a new rootkit detection method to detect rootkits based on hidden processes. We designed a detection method based on cross-view, which detects hidden process by comparing process lists attained from OS high-level and low-level respectively. The low-level process list was attained by scanning memory to find kernel object in Windows kernel, which the high-level list is attained using Windows APIs. We implemented a Windows rootkit detecting tool named VITAL to implement the proposed method and used some representative rootkits in experiments to verify the effectiveness of the proposed method.

Keywords Rootkit, Rootkit detection, Hidden process

随着网络应用的发展,网络和计算机的安全问题面临着越来越多的威胁和挑战。自从第一批针对DOS的病毒出现之后,网络上出现了特洛伊木马、蠕虫、后门等恶意软件,不断地危害着计算机系统安全和人们的信息安全。在20世纪90年代初,Internet上出现了一种新的恶意软件——rootkit[1],这是一种被入侵者或恶意软件用来隐藏踪迹的软件,通过使用rootkit,入侵者或恶意软件使得操作系统管理员不能及时感知到入侵的存在,从而可以对系统造成更严重和持久的破坏。

相对于传统的病毒检测,对rootkit的检测具有特殊的困难。传统的病毒检测使用特征码扫描的检测技术,但是rootkit为了达到隐藏的目的,不会无限制地传播,使得这种技术对于rootkit的检测很难奏效;其次,目前流行的大部分rootkit运行在操作系统的内核态,这也提高了传统方法检测rootkit的难度。因此,对于rootkit及其检测的研究具有很重要的意义,也是当前计算机系统安全研究的热点之一。另一方面,由于Windows的广泛使用,使得它成为了遭受rootkit攻击最严重的操作系统之一。尽管微软公司加大了rootkit防护的力度,仍然未能彻底抵制rootkit的侵害[2],而目前有效的Windows rootkit检测方法仍旧缺乏,所以找到一种有较强功能的Windows rootkit检测方法将具有很重要的意义。

本文选取Windows系统上的rootkit作为研究对象,首先对Windows rootkit的隐藏技术以及rootkit检测技术进行研究,以此为基础,从rootkit的行为入手,提出基于隐藏进程检测进行rootkit检测的机制,并设计了一种基于交叉视图(Cross-View)的隐藏进程检测方法。这种方法通过比较从系统高层和底层获得的进程列表,从中检测出被rootkit隐藏的进程。其中,系统高层的进程列表通过调用Windows API获得;而系统底层的进程列表通过在Windows内核中扫描内存查找EPROCESS内核对象的方法获得[3]。由于进程需要被操作系统内核调度执行,其在系统内核中的某个层次一定是可见的,而本文的方法是在获取底层进程列表时,深入到

到稿日期:2009-01-16 返修日期:2009-03-12 本文受国家科技支撑计划(No. 2006BAH02A02),国家高技术研究发展计划(863)(No. 2006AA01Z175,2006AA01Z189, 2007AA01Z462,2007AA010304,2008AA01Z133)资助。

Windows 内核中，通过查找和进程一一对应的 Windows 内核对象——EPROCESS 块获取进程列表。由于要深入到操作系统的底层，因此需要通过内核调度模块调度执行的进程都可以通过这种方法获得。

本文的检测方法获取系统底层进程列表的方式是对已有的 rootkit 检测方法的改进，从 Windows 内核的视角进行 rootkit 检测，具有较强的检测能力。使用这种方法实现了一个 Windows rootkit 检测工具 VITAL(Valorous wIndows rooTkit pALadin)，通过实验以及同其它检测方法的比较证明这种方法具有较强的 rootkit 检测能力。

1 Windows Rootkit 概述

1.1 Rootkit 定义

Rootkit 是一种在操作系统中用来隐藏进程、文件、系统数据等信息的工具包、库或者程序代码[4]。Rootkit 大多被入侵者或恶意软件用来隐藏或伪装其使用的文件、目录和进程。一旦入侵者或恶意软件使用了 rootkit，操作系统用户和管理员将很难及时感知到入侵的存在，使得他们能够达到隐藏其在被入侵系统上踪迹的目的，对系统造成更严重和持久的破坏。

Rootkit 的危害性比普通病毒更大，一方面在于使用了 rootkti 的恶意软件可以长久地存在于系统中，进行更多的破坏活动；另一方面，即使用户感知到了 rootkit 的存在，由于其隐蔽性，清除这些 rootkit 也会更加困难。

1.2 Rootkit 分类

根据 rootkit 运行的环境来分，可以把 rootkit 分为用户级、内核级、虚拟级和固件级。

1.2.1 用户级

用户级 rootkit 是指运行在操作系统用户态下的 rootkit。用户级 rootkit 主要是对用户空间下的应用程序和系统应用进行修改，包括修改应用程序的二进制文件、系统的 DLL 文件，以及修改用户空间下的一些跳转表等。用户级 rootkit 相对容易检测，因为只需要查看某一个应用程序或文件是否和最初的版本相同就可以了。

1.2.2 内核级

内核级 rootkit 是指运行在操作系统内核态下的 rootkit。大多数操作系统允许用户加载代码到内核中运行，比如 Linux 下的 LKM(Loadable Kernel Modules，可导入内核模块)，Windows 下的驱动程序，这就为 rootkit 导入到内核提供了可能。

内核级 rootkit 的危害比较大。一方面，内核级 rootkit 运行在内核中，可以随意地对操作系统进行修改，较容易地隐藏与其有关的信息；另一方面，内核级 rootkit 和操作系统内核运行在同样的级别，能够修改或破坏操作系统内核以及运行在操作系统上的检测工具，使得它们难以被检测工具检测到。

1.2.3 虚拟级

虚拟级 rootkit 是目前一种新型 rootkit，Samuel King 等人最早提出了这种设想[5]。虚拟级 rootkit 在加载后会在操作系统之下、硬件之上安装一个虚拟机，从而使得自身运行在比操作系统内核更高的级别上。Joanna Rutkowska 在 2006 年 Black Hat 会议上展示了一个虚拟级 rootkit 的原型——Blue Pill[2]。

1.2.4 固件级

固件级 rootkit 主要是指运行在 BIOS 中的 rootkit，这种 rootkit 更接近系统的底层。黑客可以利用电源配置和电源接口(ACPI) 向 BIOS 中加载 rootkit 代码。在 2006 年 Black Hat 会议上，John Heasman 演示了两个固件级的 rootkit——ACPI rootkit[6]和 PCI rootkit[7]。固件级 rootkit 是目前最接近系统底层的 rootkit，由于其位置的特殊性，重新安装操作系统甚至格式化硬盘也不会对它有任何影响。

1.3 Rootkit 技术

1.3.1 钩子技术

钩子技术的目标是替换正常的程序为 rootkit 自身的程序，从而在其中过滤由系统返回的信息。按照钩子运行的环境，可将钩子分为用户空间钩子和内核钩子。

用户空间钩子主要对应用程序和系统 DLL 本身或者它们使用的跳转表(如 IAT 和 EAT)进行修改。Rootkit 用自己的程序和 DLL 文件替换系统中的程序和 DLL 文件，或者将跳转表中的地址修改为自己提供的非法函数的地址，当调用这些函数时，执行的是 rootkit 提供的函数，使得 rootkit 可以在其中过滤掉 rootkit 需要隐藏的信息。

内核钩子主要修改内核中的一些跳转表，如 IDT，SSDT，IRP 函数表，这些跳转表的表项是一些系统服务或者回调函数的地址，rootkit 将这些跳转表的表项修改为自己提供的函数的地址。

1.3.2 内存补丁

内存补丁的技术是找到一个程序导入到内存中的位置，然后使用 rootkit 的代码覆盖内存中的区域，使得调用这段代码实际上运行的是 rootkit 的代码。内存补丁的技术出现较早，被软件破解和盗版广泛使用，但却是 rootkit 使用的先进技术之一，使用这种技术可以构造出难以检测的 rootkit。

1.3.3 直接内核对象操纵

Windows 操作系统中的任务管理器、资源管理器等与用户交互的工具通过一定的方式访问这些与之有关的对象以获得系统信息，所以 rootkit 经常会修改内核对象来达到隐藏的目的[8]。通过修改内核对象方法可以隐藏如进程、设备驱动程序、网络端口等。例如，FU rootkit 把某个进程的 EPROCESS 从双向链表 ActiveProcessLinks 中摘除，以达到隐藏进程目的[9]。

2 Windows Rootkit 的检测方法总结

2.1 特征检测法

该方法通过周期性地扫描内存和文件系统，查找是否有符合已知特征的 rootkit 被安装在系统中。这种方法目前被杀毒软件广泛使用，它的优点是便于实现，但是只能检测已知的 rootkit；另外，内核级的 rootkit 可以在内核中对扫描操作进行干扰，导致检测失败。

2.2 地址分析检测法

该方法较多地用于查找钩子。该方法分析程序中的分支(由 call 或 jmp 等指令产生)和跳转表(如 IDT 和 SSDT 等)中的地址是否位于可接受范围之内。而定义一个可接受的范围一般来说是可以完成的，比如 SSDT 表项的地址必须位于 ntoskrnl. exe 这个模块中。地址分析法是检测钩子非常有用的

方法，缺点是确定地址是否在可接受范围之内是比较困难的，特别是像 IDT 这种允许用户添加处理函数的跳转表。

2.3 执行路径分析法

执行路径分析技术的主要思想是基于如下事实：如果 rootkit 通过改变某些函数的执行路径来隐藏了一些对象，那么系统在调用被 rootkit 修改的函数时，将会执行额外的指令。执行路径分析法通过比较未安装 rootkit 系统和待检测系统上运行某个系统调用的指令数差别来分析是否被安装了 rootkit。这种方法存在一定的误差，并且不能用于分辨和确认 rootkit 的种类。

2.4 交叉视图检测法

交叉视图法从系统不同层次获得进程、文件和端口等信息，然后比较不同层次的信息的差异，这个差异可能就是 rootkit。交叉视图法是目前较常使用的 rootkit 检测方法。其核心在于如何从相对可信的系统层次获得进程、文件等信息，只要能从系统足够底层获取到信息，交叉视图法就是有效的。本文提出的 Windows rootkit 检测方法，就是一种基于交叉视图的方法。

3 一种基于交叉视图的检测方法

3.1 检测方法的提出

Rootkit 被安装到系统之后，为了隐藏入侵者的踪迹，必然会表现出一定的行为，rootkit 的行为通常是隐藏某些与之相关的进程、文件、注册表项等信息，只需要检测出进程、文件等被 rootkit 隐藏的对象，就能检测到 rootkit 在系统中的存在，所以提出了从 rootkit 的行为入手进行 rootkit 检测的机制。而在检测对象的选择上，考虑到当恶意软件入侵系统后，大多都会创建一些进程在系统中运行，因此大多数 rootkit 都会把隐藏进程作为其主要的目的，所以选择通过检测系统中的隐藏进程进行 rootkit 检测。

一般来说，内核会向用户态下的应用程序提供一些接口，使应用程序可以获得进程列表、文件列表等信息，Windows API 也是通过这些接口获得进程、文件等信息。这些信息从系统内核传递到用户态，需要经过很长的调用路径，rootkit 只需要在传递路径的某个点进行过滤就能达到隐藏的效果，路径越长，被篡改的可能性就越大，所以在系统高层下通过 API、系统命令、任务管理器、资源管理器等方式获得的信息可能是不真实的。而进程、文件等对象要在系统中存在，对于操作系统内核的某个部分一定是可见的，比如进程需要被操作系统内核进行调度执行，其对调度模块必须可见。所以，从操作系统内核足够底层的视角获得的信息相对于系统高层获得的信息更加真实。通过比较两个层次获得的信息的差别，发现被 rootkit 隐藏的对象，这就是交叉视图法的原理。

本文提出的检测方法是一种基于交叉视图的 rootkit 检测方法。首先从系统底层获得在系统中运行的进程列表，再从系统高层获得进程列表，然后比较这两个列表中的差别，如果某些进程在底层列表中存在，但是在高层列表中无法观察到，那么就有理由怀疑这个进程被 rootkit 隐藏。

3.2 相关知识

3.2.1 内核对象

在操作系统的运行过程中，需要维护一些数据结构，操作系统会在内存中存储这些信息，这些信息通常采用结构或对象的形式，这就是内核对象。Windows API 及操作系统中的任务管理器、资源管理器等与用户交互的工具通过一定的方式访问这些与操作系统中进程、文件、驱动程序等系统信息有关的对象以获得这些系统信息。

3.2.2 EPROCESS 块

EPROCESS 块是 Windows 操作系统为了管理进程所使用的内核对象，每一个进程都有一个 EPROCESS 块与之一一对应[10]。EPROCESS 块包含了进程的 PID、进程名、创建时间、PEB 等属性，以及某些和该进程相关的数据结构的指针（在 WinDbg 工具中输入命令 dt nt! _EPROCESS 可以查看其结构）。

3.2.3 DeviceIoControl

DeviceIoControl 接口是一种为 Win32 程序提供的调用驱动程序内部函数的方法。应用程序调用 DeviceIoControl API，Windows 向驱动程序发送 w32_deviceIoControl 消息，当驱动程序收到 w32_deviceIoControl 消息时，它的寄存器设置了一些值，其中包含控制码，然后驱动程序根据不同的控制码进行相应的操作，包括实现数据交换[11]。控制码是一个 32 位的整型数，其结构为：

Device Type		Required Access		Function Code		Transfer Type	
31	16	15	13	12	2	1	0

其中，最低的 0－1 两位表示的是数据传递类型；第 2－12 位是需要驱动程序执行的操作编码，一个驱动程序可能向用户态提供多种操作，这几位就是用来区别这些操作的；第 13－15 三位定义了打开 IOCTL 的方式；最高的 16－31 这 16 位用来区别不同的驱动程序类型。

3.3 检测方法的思路及关键点

本文提出的通过隐藏进程检测进行 rootkit 检测的方法是一种基于交叉视图法的 rootkit 检测方法。该检测方法分别从系统的高层和底层获取当前在系统中运行的进程的列表，然后比较两个列表的差别，在系统底层列表出现而没有在高层列表中出现的进程就是被 rootkit 隐藏的进程。

系统高层的信息在操作系统用户态下，调用 Windows API 获得，而系统底层的信息在操作系统内核态下获得。这种检测方法如图 1 所示。

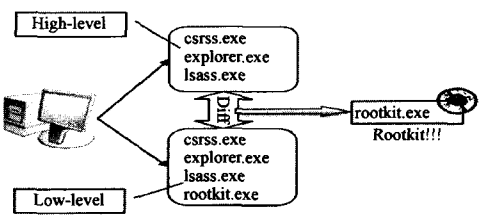

图 1 基于交叉视图的隐藏进程检测方法图示

本文提出的检测方法主要有几个关键点：

1）获取系统高层进程列表

一般来说，在系统高层获取进程信息只需要通过操作系统提供的 API 即可，Windows 操作系统提供了 NtQueryInformationProcess 等 API 可以获得当前系统中运行的进程。

2）获取系统底层进程列表

交叉视图法的核心在于如何从系统底层获取信息，接近系统底层的程度，决定了检测方法的能力。

在 Windows 中，每一个进程和内存中的 EPROCESS 内

核对象是一一对应的，找到了 EPROCESS 块就找到了进程。系统中所有的 EPROCESS 块存放在内核使用的内存中，对文献[3]中的方法进行了改进并应用在 XP sp2 和 Vista 操作系统中，通过扫描内核内存，找到所有的 EPROCESS 块，从而获得系统的进程列表。

在内存中，所有的 EPROCESS 被组织成一个双向链表，可以通过遍历链表的方式获得 EPROCEE 块[12]，但是考虑到存在 FU rootkit 这类把某个进程的 EPROCESS 从双向链表中摘除达到隐藏效果的 rootkit，使用扫描内存的方法将更加有效。

3)进入操作系统内核

检测方法在内核中扫描内核使用的内存，需要访问内核的页目录和页表，这在用户态是无法完成的，所以需要将获取系统底层列表的程序加载到内核中运行。

在 Windows 操作系统中，服务和驱动程序是运行在系统的内核态的，同时，Windows 也为加载驱动程序到内核提供了接口。将检测方法中获取系统底层进程列表的部分实现为驱动程序的形式，导入到内核中运行，在内核中获取进程列表后，通过 DeviceIoControl 接口将获得的信息传送至用户态。

3.4 检测方法的实现

3.4.1 获取高层信息

使用 CreateToolhelp32Snapshot，Process32First 和 Process32Next等 Windows API 获取系统高层信息。

(1)使用 API CreateToolhelp32Snapshot 获得系统快照。这个 API 对指定的进程、堆、模块、线程建立一个快照，并返回这个快照的句柄；

(2)使用 Process32First 和 Process32Next 访问上一步中获得的快照，得到系统中进程的列表。

3.4.2 获取底层信息

1)进入系统内核

由于系统中的所有 EPROCESS 块都存放在内核地址空间，要访问它们需要在内核中完成，所以需要将获取系统底层进程列表的程序加载到内核中运行。我们将这部分程序实现为驱动程序的形式，加载到内核中执行。

在本文的检测方法中，加载驱动程序的方法是使用服务控制管理器(Service Control Manager，SCM)，加载的流程是：

(1)使用 OpenSCManager 获得服务控制管理器的句柄；

(2)使用 Windows API CreateService 向系统添加一个服务；

(3)调用 StartService 开始运行服务；

(4)使用 CreateFile 打开驱动程序，并获得它的句柄。

将驱动程序导入到内核并获得它的句柄后，就可以使用 Windows API DeviceIoControl 向驱动程序发送 I/O 控制码，向驱动程序发出命令，并得到驱动程序通过缓冲区发送回来的系统底层信息。

2)扫描内存中的 EPROCESS 获取底层进程列表

系统中所有进程的 EPROCESS 块都存放于 MmSystemRangeStart 至 System 进程对应的 EPROCESS 所在地址之间。其中 MmSystemRangeStart 是一个内核导出的常量，表示内核内存开始的地方，也就是 0x80000000；获取 System 进程对应 EPROCESS 地址的方法将在 3.5.1 节中给出。获得进程列表的步骤如下。

(1)依次枚举从 0x80000000 到 System 进程对应 EPROCESS 块之间的地址，扫描内存的方法在 3.5.2 节中给出；

(2)检查当前地址是否在内存中，如果是，将该地址的内容作为一个 EPROCESS 结构；

(3)通过将在 3.5.3 节中描述的方法判断该 EPROCESS 结构是否合法，如果合法，记录结果；

(4)将地址设置加上一个 EPROCESS 结构的大小，跳过此不合法的 EPROCESS，继续扫描。

3.5 关键步骤分析

3.5.1 获取 System 进程的 EPROCESS 地址

Windows 中所有驱动程序都是通过 System 进程加载的，它们都作为 System 进程的线程在系统中运行，所以 System 进程对应 EPROCESS 的地址，可以在驱动程序中调用 PsGetCurrentProcess()获取。

3.5.2 扫描内存的实现

在获取内存的过程中，需要检查地址是否在内存中，文献[3]中的方法是通过查看 PTE 和 PDE 的有效性，这涉及到 Windows 虚拟地址到物理地址的转换机制，从一个虚拟地址得到 PTE 和 PDE 的方法是：

```
pde = (VirAddr >> 22) * 4 + PageDirAddr;
pte = (VirAddr >> 12) * 4 + PageTableAddr;
```

其中，页目录和页表地址在 WinDbg 中通过命令！pte 可以获得，在 XP SP2 中分别为 0xc0300000 和 0xc0000000。PDE 和 PTE 的最后一位都是 Present 位，只需要判断该位即可判断一个虚拟地址是否有效。

但是，通过在 Windows Vista 系统上进行实验发现，这种方法太过接近系统底层，而由于物理地址扩展(Physical Address Extension，PAE)机制的引入，在 Windows Vista(还可能包含其他版本的 Windows 操作系统)中无法实现。于是，直接使用 Windows DDK 提供的 MmIsAddressValid()函数来判断内存地址的有效性。

3.5.3 判断 EPROCESS 的有效性

一个内核对象中会包含一个标志其类型的结构 OBJECT_HEADER(在调试工具 WinDbg 中通过 nt! _OBJECT_HEADER 可以查看其结构)，该结构中有一个指针 Type 指向了一个标识对象类别的结构体，所有 EPROCESS 外层的 OBJECT_HEADER 结构中，Type 值是相同的，以此作为判断一个内存地址是否指向一个 EPROCESS 内核对象的依据。在 3.5.1 节中已经获得了 System 进程的 EPROCESS，可以以其 Type 值作为参考标准。

操作系统将所有进程的 EPROCESS 结构都存放在内存中，当进程退出后也不会主动释放它的 EPROCESS 所占用的内存，所以搜索到的进程包括已经退出运行的进程，需要通过一定的方式确定搜索到的 EPROCESS 的有效性。

在 WinDbg 查看 EPROCESS 的内容，在该数据结构中，被关注较多的主要有以下几项(在 Windows XP sp2 下的内容)：

```
lkd> dt nt! _EPROCESS
+0x078 ExitTime   : _LARGE_INTEGER
   ......
+0x084 UniqueProcessId   : Ptr32 Void
+0x088 ActiveProcessLinks   : _LIST_ENTRY
   ......
```

```
+0x174 ImageFileName  : [16] UChar
    ......
+0x1b0 Peb  : Ptr32 _PEB
    ......
```

其中 ExitTime 可以作为判断 EPROCESS 是否合法的依据：ExitTime 是进程的退出时间，对于所有退出运行的进程来说，这一项都是 0。

4 实现及实验分析

4.1 工具实现

目前，我们已经根据本文提出的基于交叉视图的进程检测方法实现了一个基于隐藏进程检测的 Windows rootkit 检测工具 VITAL。工具采用了基于 DLL 插件机制的结构，分为 GUI 层、DLL 层和内核层，插件结构使得检测工具便于升级，层次结构使得工具有一部分运行在 Windows 内核中，提高了检测的能力。

该检测工具在 Windows XP SP2 下开发完成，并在 Windows Vista ultimate 操作系统上完成部分测试。GUI 层和 DLL 层使用 VC 6.0 作为开发工具，使用 Windows DDK 2600 进行内核层的开发，内核层由若干驱动程序组成。在开发中，使用了 Windbg 和 IDA 等工具进行辅助。

4.2 实验及分析

选用 Windows XP Professional SP2 作为实验平台，选用 Windows 操作系统上比较典型和常见的几个 rootkit 进行测试，包括 Hacker Defender，NtRootkit，HideProcessHook 以及 FU Rootkit，同时，选取 Windows 工具 Processes Explorer 和 rootkti 检测工具 Rootkit Unhooker[13] 作为对照。各个 rootkit 的原理及实验结果如表 1 所列。

表 1　VITAL 和部分工具的检测结果对比

	Process Explorer	Rootkit Unhooker	VITAL
Hacker Defender	×	√	×
NtRootkit	×	√	√
HideProcessHook	×	√	√
FU Rootkit	×	×	√

Process Explorer 是微软公司提供的查看 Windows 上正在运行的进程列表的工具，其原理和 Windows 自带的任务管理器相似，都是通过调用系统 API 来查看进程列表，没有 rootkit 检测的功能，所以 rootkti 一般都能对其达到隐藏的效果。Rootkit Unhooker 采用钩子检测的方式进行 rootkti 检测，这种工具能检测大部分用户态 rootkit 以及使用 SSDT 钩子进行进程隐藏的 rootkit，但是不能检测到类似 FU Rootkit 这种使用直接内核对象操纵的 rootkit。Rootkit Unhooker 是一种从 rootkit 使用的技术入手进行 rootkit 检测的方法，由于 rootkit 使用的技术较多，变化较快，使得这种方法具有一定的局限；而不管 rootkit 使用何种技术，它表现的行为都是隐藏，本文的方法从 rootkit 的行为入手，取得了比较好的效果。

结束语　本文以对 Windows rootkit 的研究为基础，提出了通过隐藏进程检测进行 rootkit 检测的机制，并设计了一种基于交叉视图的隐藏进程检测方法，最终实现了一个 Windows rootkit 检测工具。通过实验，验证了该工具具有较强的检测能力，在某些方面优于目前出现的检测工具。

随着 Internet 的进一步发展，新的 rootkit 将会不断涌现，根据目前对于 rootkit 发展趋势的研究，我们认为进一步的工作将会包含这些内容：提出功能更强的检测方法，增加钩子、文件、注册表项等的检测以完善检测工具的能力；对 Windows 新推出的操作系统如 Windows Vista 提供 rootkit 检测；对 rootkit 监测技术进行研究，变事后发现为事先监测；引入虚拟机等技术对检测工具进行保护，防止 rootkit 对其的破坏。

参 考 文 献

[1] CERT ® Advisory CA-1994-01 Ongoing Network Monitoring Attacks[OL]. http://www.cert.org/advisories/CA-1994-01.html

[2] Rutkowska J. Subverting Vista™ Kernel For Fun And Profit [J]. Blackhat Presentation, August 2006

[3] Uty. 搜索内存枚举进程[OL]. http://blog.donews.com/uuty/archive/2006/03/15/769472.aspx

[4] Wiki[OL]. http://en.wikipedia.org/wiki/Rootkit

[5] King S T, Chen P M. SubVirt: Implementing malware with virtual machines[C]// Security and Privacy, IEEE Symposium. 2006

[6] Heasman J. Implementing and Detecting an ACPI Rootkit[M]. BlackHat Federal, 2006

[7] Heasman J. Implementing and Detecting a PCI Rootkit[M]. November 2006

[8] Hoglund G, Butler J. Rootkits: Subverting the windows kernel. 2007

[9] FUrootkit[OL]. http://www.rootkit.com/vault/fuzen_op/FU_Rootkit.zip

[10] Russinovich M, Solomon D. Microsoft Windows Internals[M]. MicroSoft Press, 2005: 289-295

[11] Baker A. Windows NT 设备驱动程序设计指南[M]. 北京：机械工业出版社，1998：44-45

[12] Jimhotkin. Detection of the hidden processes[OL]. https://www.rootkit.com/newsread.php? newsid=434

[13] Rootkit Unhooker[OL]. http://www.antirootkit.com/software/RootKit-Unhooker.htm

(上接第 88 页)

[3] Open Geospatial Consortium, Inc. Vision and Mission. WWW document[OL]. http://www.opengeospatial.org/about/?page=vision, 2006

[4] Michaelis D C, Ames P D. Evaluation and Implementation of the OGC Web Processing Service for Use in Client-Side GIS. Geoinformatica DOI 10.1007/s10707-008-0048-1

[5] Open Geospatial Consortium, Inc. OpenGIS ® Web Processing Service (WPS) Specification. WWW document[OL]. http://portal.opengeospatial.org/files/? artifact_id=24151, 2007

[6] Kim Hak-Hoon, LEE Kiwon. Web-based GIS-transportation framework data services using GML, SVG and X3D. Dept. of Information System Engineering, Hansung University Seoul, Korea: 136-792

[7] 何强，马颂德. 图像镶嵌技术理论、难点及应用[J]. 高技术通讯，1998(3)

[8] Web Processing Service (WPS). WWW document[OL]. http://52north.org/index.php? option = com_projects&task = showProject&id=21&Itemid=127

[9] http://www.jump-project.org/

[10] Standards for XML and Web Services Security [J]. Martin Naedele, Computer, 2003, 36(4): 96-98