doi 10.3969/j.issn.1671-1122.2014.02.008

一种基于 Windows 内核驱动的可疑样本采集系统的设计与实现

张涛[1]，焦英楠[2]，禄立杰[1]，文伟平[1]

（1. 北京大学软件与微电子学院，北京 102600；2. 国家计算机网络应急技术处理协调中心，北京 100029）

摘 要：文章研究一种结合规则库扫描和基于 Windows 内核驱动的程序行为分析的可疑样本采集系统，将大大提高样本采集的全面性和准确性，对加快病毒的发现和病毒库的更新具有重要意义。文章首先分析 Windows 操作系统的体系结构，接着给出了基于 Windows 内核驱动的可疑样本采集系统的整体架构，最后根据系统架构对各个模块进行详细设计和实现，并给出了一个测试用例及结果分析。实验结果表明，该系统能够准确、高效地采集可疑样本信息。

关键词：内核驱动；可疑样本采集；规则库

中图分类号：TP309 文献标识码：A 文章编号：1671-1122（2014）02-0041-07

The Design and Implementation of Suspicious Sample Collection System based on Windows Kernel Driver

ZHANG Tao[1],JIAO Ying-nan[2],LU Li-jie[1],WEN Wei-ping[1]

(1.*Department of Information Security, SSM, Peking University, Beijing* 102600, *China*;2.*National Computer Network Emergency Response Technical Team / Coordination Center, Beijing* 100029, *China*)

Abstract: The study of suspicious sample collection system with the rule-based scanning and procedures behavior analysis based on Windows kernel driver will greatly enhance the comprehensiveness and accuracy of sample collection, and it has important significance to accelerate the discovery of the virus and the virus database updates. Firstly, this paper analyzes the architecture of Windows operating system and then gives the overall system architecture of the suspicious sample collection system based on Windows kernel drivers. Finally, according to the system architecture, the paper detailed designs and implements of each module, and gives examples and the results of a test. The experiment shows that the system is capable of accurately and efficiently collect samples of suspicious information.

Key words: kernel driver; suspicious sample collection; rule base

0 引言

随着计算机网络技术的不断发展，人们在享受高速网络提供的便捷服务的同时，也面临着越来越多的网络安全问题和安全威胁。面对网络上数量如此庞大的病毒和木马程序，我们目前主要依靠安全厂商的杀毒软件产品进行防御和杀毒。但是这些安全产品大多依靠于病毒库进行查杀，而病毒库需要等待病毒较大规模发作、安全厂商捕获到病毒和木马样本，并经分析后才能更新，需要一段时间才能完成。而这段时间内用户的电脑仍然面临着威胁，特别是零日漏洞的安全威胁。

目前已经采用可疑样本采集系统的厂商，检测可疑样本的方式仍然是通过病毒库的方式进行全盘扫描，在扫描后找出一些可疑的程序样本，但还不能确定这些是否是木马或者病毒的样本程序。显然，这种检测方式依然存在依赖病毒库时效性的问题。

因此，研究一种结合规则库扫描和基于 Windows 内核驱动的程序行为分析相结合的可疑样本采集系统，将大大提高可

收稿日期：2013-08-23

基金项目：国家自然科学基金 [61170282]

作者简介：张涛 (1986-)，男，江西，硕士研究生，主要研究方向：漏洞分析和漏洞挖掘；焦英楠（1983-），女，辽宁，工程师，硕士，主要研究方向：软件工程、信息安全等；禄立杰 (1974-)，男，河北，硕士研究生，主要研究方向：软件工程；文伟平 (1976-)，男，湖南，副教授，博士，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等。

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

疑样本采集的全面性、准确性和时效性，对加快病毒和木马程序的发现和病毒库的更新，保障用户的安全具有非常重要的意义。

1 Windows 内核驱动相关技术

1.1 内核驱动技术

Windows 操作系统分为用户模式和内核模式，一般的用户进程运行在用户模式，驱动程序和内核组件运行在内核模式。Windows 内核驱动程序依据其用途不同，主要分为即插即用驱动程序、内核扩展驱动程序和文件系统驱动程序三大类。其中，即插即用驱动程序通常是为了驱动硬件设备而由硬件厂商提供，与 Windows 的 I/O 管理器、即插即用管理器和电源管理器一起工作 ；内核扩展驱动程序扩展内核的功能，提供了访问内核模式代码和数据的途径 ；文件系统驱动程序接收针对文件的 I/O 请求，再进一步将这些请求转变成真正对于存储设备或网络设备的 I/O 请求，从而满足客户的原始请求。

内核驱动程序不同于用户模式的应用程序，通常来说，内核驱动程序设置了一系列回调函数集合，这些回调函数在内核驱动程序中被称为例程，它们被操作系统的 I/O 管理器在适当的时候调用。

根据驱动程序的类型，I/O 管理器在如下所示的条件出现时，就会调用驱动程序例程，这些条件有 ：装入驱动程序、卸载驱动程序、插入或移走设备、用户模式程序发出一个 I/O 系统服务调用、驱动程序能够使用共享硬件资源以及在真实设备运行期间的各个关键时刻。

内核驱动程序有一些关键的数据结构，包括驱动程序对象、设备对象以及 I/O 请求包，现对其进行简要描述 ：

1）驱动程序对象

每个驱动程序只有唯一的一个驱动程序的对象。当装入一个驱动程序时，I/O 管理器建立一个驱动程序对象。如果驱动程序初始化失败，I/O 管理器删除该对象。在 DriverEntry 初始化的过程中，直接设置各派遣例程的入口地址到驱动程序对象的各个域里面。

2）设备对象

I/O 管理器和驱动程序在任何时刻都需要知道 I/O 设备所进行的工作。设备对象通过保存关于设备特征和状态的信息，使上述任务成为可能。系统上的每个虚拟、逻辑和物理设备有一个设备对象。

3）I/O 请求包（IRP）

IRP 是一个内核“ 对象 ”，它是一个预先定义的数据结构。带有一组对它进行操作的 I/O 管理器例程。I/O 管理器接收一个 I/O 请求，然后在把它传递到合适的驱动程序堆栈中的最高驱动程序之前，分配并初始化一个 IRP。

一个 IRP 有一个固定的首部和可变数目的 IRP 堆栈单元块。每个 I/O 请求有一个主功能代码（如 IRP_MJ_CREATE 对于文件打开）并可能有次功能代码。IRP 的固定部分（IRP 结构自身）含有 IRP 的固定属性，每个堆栈单元（一个 IO_STACK_LOCATION 结构）事实上含有大多数有关的 IRP 参数。当一个 IRP 由多个驱动程序处理时，使用多个 IRP 栈单元。每个驱动程序从当前 IRP 栈单元得到它的 IRP 参数。

1.2 SSDT HOOK驱动

SSDT（System Service Dispatch Table，系统服务描述符表）用来查询处理系统调用的特定函数。也就是说，这个表把 ring3 下 UserMode 的 Win32 API 同 ring0 下 Kernel API 相联系。

SSDT 通过索引系统调用号来查找在内存中的函数地址。而另一个被称为 SSPT（System Service Parameter Table，系统服务参数表），它则指定了每一个系统服务的函数参数的字节数。KeServiceDescriptorTable 是一个内核的导出表，这个表中包含一个指向部分 SSDT 的指针。而 SSDT 中则包含着内核的主要部分，Ntoskrnl.exe 中的核心系统服务的实现。

通过修改 SSDT 中的函数入口地址来 HOOK SSDT 中的函数，可以过滤掉你所关心的特定结果，从而欺骗操作系统。比如，你可以隐藏特定的进程，隐藏文件，隐藏端口等。

1.3 TDI驱动

TDI（Transport Driver Interface，传输层接口）使网络 API 驱动程序可以按照 TDI 接口来调用更低层的协议驱动程序，消除了网络 API 与低层的传输协议之间的紧耦合关系。按照 TDI 接口来实现的传输协议可以被多个 TDI 客户使用。

在 TDI 这一层，网络 API 的内核驱动程序是 TDI 客户，

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

而协议驱动程序则是 TDI 传输器。网络 API 驱动程序接收各种形式的网络请求，例如，名称解析、建立连接、发送或接收数据等，TDI 使得这些网络请求被统一描述，因而各种传输协议只要按照 TDI 的规范来实现，就可以被网络 API 使用。

1.4 NDIS中间层驱动

NDIS（Network Driver Interface Specification，网络驱动程序接口规范）为传输层提供了标准的网络接口。在 Windows 操作系统中，所有的应用程序都最终通过调用 NDIS 接口，实现网络访问。

NDIS 支持三种类型的驱动程序 ：

1）协议驱动程序 ：对上层应用程序开放 TDI 接口，对下层驱动程序开放 Protocol 接口，与下层 Miniport 接口对接，实现协议驱动功能。

2）中间层驱动程序 ：位于物理网卡驱动程序（即微软官方所称的微型端口驱动程序）之上，同时具备 Miniport 接口和 Protocol 接口，可与上、下层驱动程序对接，提供转发驱动功能。

3）微型端口驱动程序 ：开放 Miniport 接口，再通过 NDIS 接口完成对物理网卡的操作，实现对物理网卡的驱动功能。

2 Windows 体系结构

2.1 Windows内核模式

内核模式的构成文件是系统的核心文件，主要包括 HAL.DLL、NTOSKRNL.EXE、设备驱动、文件系统驱动、图形设备驱动、WIN32K.SYS 六个部分。

其中，HAL（硬件抽象层）使得 reactOS 内核可以运行在不同的 X86 母板上。HAL 为内核抽象母板的特定代码，是对不同母板定义一种抽象的接口，并向上提供一种标准的接口调用。

NTOSKRNL 又分为两层，第一层称为核心层（CORE），提供非常原始且基本的服务，如多处理器的同步、线程调度、中断分派等，第二层是执行体（EXECUTIVE），内核执行体提供了系统的服务，这里的服务指的是系统函数，这些函数可以被划分为具备虚拟存储的内存管理、对象管理、进程线程管理、配置管理、安全引用监视、I/O（输入 / 输出）管理、PNP（即插即用管理器）、电源管理、缓冲管理器以及本地过程调用（LPC）管理等多个类别。

设备驱动程序是核心态可加载模块（以 .SYS 为扩展名，存放在 %SYSTEMROOT%SYSTEM32\DRIVERS 目录），它们是 I/O 管理器和相关硬件设备的接口。

文件系统驱动程序也是核心态可加载模块（以 .SYS 为扩展名，存放在 %SYSTEMROOT%SYSTEM32\DRIVERS 目录），文件系统其实是强加给存储硬件的一种文件存放规则。某类文件系统其实就是按照它的文件存取规则在存储器上组织文件的信息。

图形设备驱动很独特，只有内核模式设备驱动 WIN32K.SYS 才能启动。

WIN32 子系统是 Windows 操作系统的内核部分（原生子系统，其他的子系统是可以分割的），如果没有这个子系统，Windows 操作系统就不能运行，原因是 WIN32 的文档化的 API 都是通过这个子系统实现的。WIN32K 又被划分成 USER32 和 GDI32 两个部分。

2.2 Windows用户模式

用户模式是 Windows 系统和使用者直接交互信息的层次，主要包含以下三个部分 ：

1）核心 DLL（动态链接库）

核心 DLL 主要包括 NTDLL.DLL、USER32.DLL 以及 KERNEL32.DLL。

2）系统核心进程

系统的核心进程主要包括 SERVICES.EXE、SVCHOST.EXE 、LSASS.EXE 和 WINLOGON.EXE 四个。

3）系统应用程序

系统应用程序主要提供管理系统的一些功能，包括进程管理器、用户控制面板和系统策略管理等多种应用程序。

3 系统设计与实现

3.1 开发环境

本系统开发过程中搭建的主要环境描述如下 ：

1）两台主机 ：一台主机（HOST），一台目标机（TARGET）

2）操作系统 ：Windows Server 2003

3）开发工具 ：Windows DDK 3790.1830，Visual Studio 2008，MS SQL Server 2008

4）调试工具 ：WinDbg 6.6，虚拟机

5）版本控制工具 ：SVN 1.6

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

3.2 系统体系

如图1所示，系统采用层次化、模块化的结构设计，具体分为内核模式的SSDT Hook驱动、TDI驱动以及NDIS中间层驱动，用户模式的数据采集系统、数据分析系统。

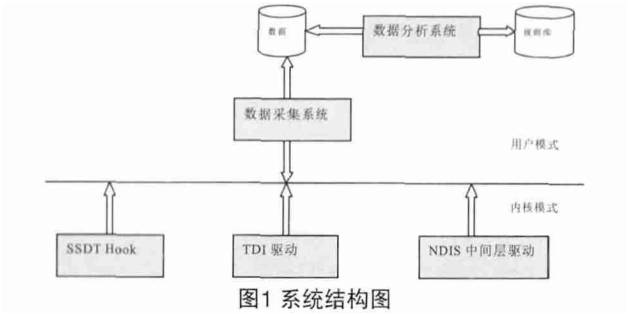

图1 系统结构图

3.3 概要设计

3.3.1 内核驱动层功能模块

在本系统中，内核驱动模块分为3个处于操作系统底层不同层次独立的驱动程序，分别是SSDT HOOK驱动、TDI过滤驱动以及NDIS中间层驱动。它们在操作系统启动的时候，大部分其他模块未加载之前及系统未登录之前就开始加载运行，以确保在木马等攻击程序加载之前运行，采集符合默认规则的数据，并写入本系统定义的可疑数据库中。

3.3.2 应用层功能模块

应用层分为两个子系统，分别是数据采集子系统和数据分析子系统。其中，数据采集子系统工作在用户模式，与内核驱动模块进行通信，包括原始数据采集、系统状态数据采集、实时行为数据采集、实体行为数据采集、实体生理数据采集5个模块。数据分析子系统也工作在用户模式，它将内核驱动模块和监控采集子系统采集的可疑数据进行清洗、转换、加载，形成适合本子系统分析以及便于查看的数据立方体，并保存到关系数据库中。再结合规则库，深入挖掘哪些数据操作是不安全的行为，并找出引发该操作的进程等信息，形成分析报告，给出服务器安全系数。

3.4 功能模块设计

3.4.1 SSDT HOOK驱动模块

SSDT HOOK驱动是本系统内核模块用来监控文件的操作、注册表的操作、进程方面的操作，并把监控结果保存到自定义的数据库中。

监控文件的操作主要是通过HOOK SSDT中关于文件方面操作的函数，对以下6个主要操作进行监控：创建文件、打开文件、读文件、写文件、删除文件以及设置文件属性。

3.4.2 TDI过滤驱动模块

TDI驱动是本系统内核模块中用来监控进程的网络通信情况，通过绑定TCP/IP协议生成的设备，可以在内核层监控到各个进程的主要网络通信协议数据包，分别是通过绑定下面3个设备进行3个主要网络协议的监控：\Device\Tcp、\Device\Udp以及\Device\IP。

3.4.3 NDIS中间层驱动模块

NDIS中间层驱动是本系统内核模块中在网络驱动底层对网络通信数据包进行监控的模块，由于本系统的TDI驱动只绑定TCP/IP协议生成的设备，而对于某些不经过TCP/IP设备的网络通信，TDI驱动就无法发挥作用了，因而为了全面监控网络通信数据包，有必要在网络驱动底层进行监控，本系统选择了在NDIS中间层进行处理。

3.4.4 数据采集模块

数据采集系统是本系统的核心子系统，其目的就是获取目标主机系统的真实环境数据。但是由于操作系统是一个基础的平台软件，其提供了进程、线程、同步机制、内存管理、对象管理等多种复杂运行机制，同时在操作系统平台上还加载和运行了大量的驱动和应用程序（木马、病毒只是其中一部分），这些特性决定了目标系统的整体运行状态和运行环境是富于动态变化和非常复杂的。

为了让用户更直观地管理和查询，该子系统也集成了以木马名称为项目的管理，并提供添加、删除和修改操作，同时也提供单独对DNS进行查询以及查看动态检测结果和静态检测结果。

3.4.5 数据分析模块

数据检测分析子系统运行在非目标系统的主机上，主要是对采集后的数据经转换形成的关系数据库进行处理，以可执行模块为中心，对各种数据分析挖掘，找出木马等有害信息。

1）数据库设计

在本子系统中，我们选用的数据库是微软新推出的功能比较强大的SQL Server 2008，它能对大数据量的信息进行有效的管理，提供高效的搜索、查询、数据分析、挖掘、报表、数据整合以及数据同步等功能。编程接口我们使用

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

的是 ADO（Active Data Objects，ActiveX 数据对象）技术，它是 Microsoft 提出的应用程序接口（API）用以实现访问关系或非关系数据库中的数据。

2）数据分析

主要分为可疑执行体快速定位和关联信息自动挖掘两个阶段：

（1）可疑执行体快速定位

对关联数据集的数据进行分析，根据相应的策略集来判断是否有可疑的行为，从而定位可疑的可执行模块，如图 2 所示。这里，检测分析子系统提供了 3 种主要的定位策略：隐藏特征、规则过滤和检查人员手工过滤。

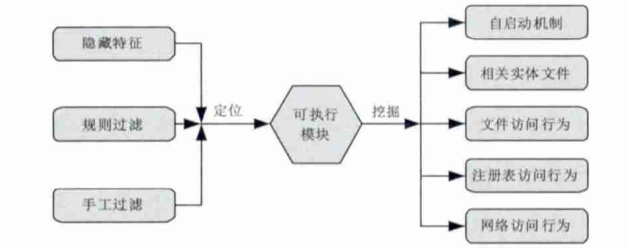

图2 以可执行模块为轴的数据关联重组

（2）关联信息自动挖掘

对该可执行模块进行关联数据挖掘，以可执行模块为线索去挖掘其全部行为特征，包括其自启动机制、相关的实体文件、文件访问行为、注册表访问行为和网络访问行为等，最终形成一个分析报表提交给用户。

对于数据检测分析子系统，其系统模块实现如图 3 所示。

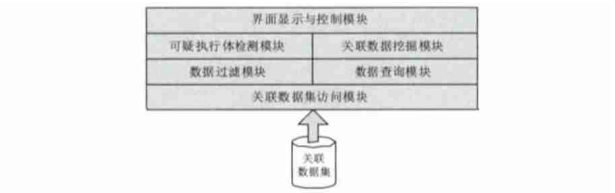

图3 数据检测分析子系统模块实现图

3.5 系统实现

3.5.1 SSDT Hook驱动实现

SSDT Hook 驱动被系统执行体 I/O 管理器调入运行后，首先执行 DriverEntry 入口点函数，初始化全局变量并修改系统 SSDT 写，Hook 需要监控的内核 SSDT 导出函数，并进行监控数据保存，修改 SSDT 内核保护的核心代码如图 4 所示。

3.5.2 TDI驱动实现

在本系统中，TDI 驱动指的是传输层接口过滤驱动，其挂接到 TDI 传输驱动程序 Tcpip.sys 创建的三个设备对象之上，截获 TDI 客户和传输驱动程序之间的 IRP 通信，解析 IRP 堆栈包中的主功能码和次功能码，并在对应的 Dispatch 例程中处理。在这些 Dispatch 例程中，可以获取通信双方的 IP 地址、端口号、协议等五元组信息。处理后再将 IRP 传递给所挂接的设备。

3.5.3 NDIS中间层驱动实现

NDIS 中间层驱动位于 NDIS 协议驱动和 NDIS 小端口驱动之间，可以截获所有的网络通信数据包，本系统只处理以太网包。对处于 TCP/IP 协议栈不同层次的常见协议数据包进行解析，监控记录满足设定规则的数据包。如在网络层分析 IP 协议数据包和 ARP 协议数据包、在传输层分析 TCP 协议数据包和 UDP 协议数据包、在应用层分析 HTTP 协议数据包和 DNS 协议数据包等。

```
    // 通过 MDL 修改只读内存
    PMDL  g_pmdlSystemCall;
    PVOID *g_ppMappedSystemCallTable;
    g_pmdlSystemCall = MmCreateMdl(NULL,
KeServiceDescriptorTable.ServiceTableBase, KeServiceDescriptorTable.
NumberOfServices * sizeof(DWORD));
    if(!g_pmdlSystemCall)
    return;
    MmBuildMdlForNonPagedPool(g_pmdlSystemCall);
    // 修改标志位
    g_pmdlSystemCall->MdlFlags = g_pmdlSystemCall->MdlFlags |
MDL_MAPPED_TO_SYSTEM_VA;
    g_ppMappedSystemCallTable = MmMapLockedPages(g_
pmdlSystemCall, KernelMode);
    // 在此 Hook SSDT 导出函数
    // 设置完毕后需要恢复内存标志
    if(g_pmdlSystemCall)
    {
    MmUnmapLockedPages(g_ppMappedSystemCallTable, g_
pmdlSystemCall);
    IoFreeMdl(g_pmdlSystemCall);
    }
```

图4 修改SSDT内核保护的核心代码

4 系统测试与分析

4.1系统测试

4.1.1 测试环境

1）数据采集端

两台主机，操作系统分别为 Windows XP SP3 系统和 Windows Server 2003 SP2 系统，硬件配置均为双核 CPU、2G 内存、160G 硬盘。

2）分析端

单台主机，操作系统为 Windows Server 2003 SP2 系统，

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

数据库为 SQL Server 2008，硬件配置为四核，4G 内存，320G 硬盘大小。

3) 木马样本

攻击样本选用了经典的“灰鸽子”木马，“威金”病毒。

4.1.2 运行效果

1) 驱动运行状态

驱动安装成功后，与本系统相关的 SSDT 函数会被 Hook，被自定义的函数所取代，如图 5 所示。

图5 SSDT函数被本系统驱动自定义的函数所取代

操作系统 TCP/IP 驱动导出的 \Device\Tcp、\Device\Udp、\Device\RawIp，已经被本系统的过滤驱动所绑定，如图 6 所示。

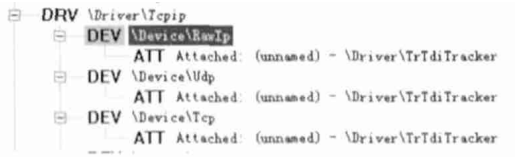

图6 TCP、UDP、RawIp被本系统的过滤设备绑定

2) 数据采集

设置数据集名称、保存的路径、已经采集规则后，通过接口层向内核层驱动模块下发指令，接收驱动模块返回的数据，并实时记录到本系统自定义的文件中。

3) 即时木马检测

在采集端实时采集时，可以通过创建项目，自定义木马病毒行为特征，来即时检测木马。图 7 显示 test 项目有包含木马行为特征。

图7 木马检测

4) 原始数据转换

采集端采集原始数据完毕后，系统是以数据集的方式，通过自定义的文件格式保存在采集端电脑的硬盘里。我们需要利用接口层的转换工具把其转换为关系数据库格式，以供数据分析系统使用。图 8 显示正在转换中的数据。

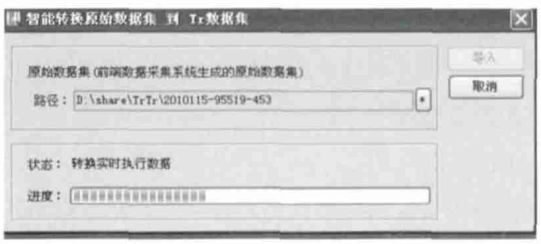

图8 原始数据转换

5) 数据分析结果

打开数据分析系统，系统为用户直观展示了转换后的数据集。如图 9 所示。

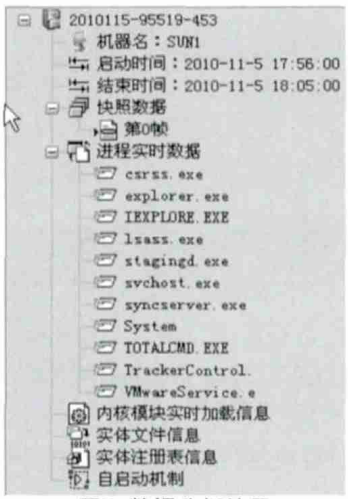

图9 数据分析结果

4.2 性能分析

4.2.1 操作系统资源使用情况

本系统的原始数据采集是由内核层驱动模块来完成的，而驱动通过线程池、内存池、缓存的方式，对操作系统资源进行了有效的管理，表 1 列出了在某个时刻监控的未安装驱动与安装驱动后的资源使用情况。

表1 操作系统资源使用情况

资源使用	驱动安装前	驱动安装后
CPU 使用	7%	9%
非分页内存使用	10%	20%
分页内存使用	15%	20%
网络带宽使用	2%	2%

从表 1 可以看出，驱动安装后，操作系统资源并未出现大规模被使用的情形，这说明系统的资源占用率是在可以接受范围之内的。

4.2.2 隐藏信息检测

在对本系统的测试中，我们设置了对 Windows 操作系统关键目录进行数据采集，如 Windows 目录、System32 目录，

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

这两个目录是许多木马、病毒程序写入数据的地方。通过对采集数据后的分析发现，在 C:\Windows 目录下存在 G_Server.exe、G_Server.dll 和 G_Server_Hook.dll 三个隐藏文件，在隐藏进程列表中，我们也发现了 G_Server.exe 这个进程被隐藏了，而这些正是“ 灰鸽子 ” 木马的典型特征，因为我们的测试样本用到了“ 灰鸽子 ” 木马，说明本系统可以准确地发现该木马。

4.2.3 可执行模块注入检测

对操作系统的关键进程，如 Explorer.exe、SVCHOST.EXE、winlogon.exe 等的加载模块进行检测，通过对比正常的加载模块，我们发现在 Explorer.exe 进程中，其加载模块包含 vdll.dll 动态链接库，而这个正是我们前面测试样本中“ 威金 ” 病毒的特征。

4.3 安全性分析

作为一个安全检测系统，需要检测木马和病毒等各种有害攻击，所以其自身的安全性至关重要。

1) 原始数据的采集是由底层的内核驱动来实现，在操作系统启动还未登录时就开始工作，可以有效检测木马隐藏的信息。

2) 采集到的原始数据以自定义的文件格式保存，避免以明文的形式存在，防止因原始数据文件丢失而产生泄露。

3) 工作在操作系统应用层的数据采集系统、数据分析系统均在内核实施了驱动级别的保护，以阻止被木马、病毒或其他恶意行为终止。

5 结束语

本文针对 Windows 内核架构模型，切实调研了现有研究成果，设计和实现了基于 Windows 内核驱动的可疑样本采集系统。本文仍有进一步的工作要做 ：在现有基础上做进一步的优化设计，将支持向量机算法应用于采集系统中，在大规模的网络环境中进一步测试、改进和优化。（责编 杨晨）

参考文献

[1] Mark E.Russinovich，David A.Solomon. 深入剖析 Windows 操作系统（第四版）[M]. 北京 ：机械工业出版社，2007.

[2] 杨蛟龙 . 基于 SSDT HOOK 技术的 Ring0 级进程保护组件设计与实现 [D]. 北京 ; 北京理工大学 ,2009.

[3] 方飞 , 李兵 .Windows 系统 TDI 驱动关键技术的研究 [J]. 通信技术 , 2010, 43(7):3.

[4] 林锐，韩永泉 . 高质量程序设计指南——C++/C 语言（第 3 版）[M]. 北京 ：电子工业出版社 ,2007.

[5] 范昕炜 . 支持向量机算法的研究及其应用 [D]. 浙江 ：浙江大学 ,2003.

[6] 百度百科 . 灰鸽子 [EB/OL].http://baike.baidu.com/view/25407.htm.

[7] 池水明，周苏杭 . DDoS 攻击防御技术研究 [J]. 信息网络安全，2012，(05)：27-31.

[8] 黄华军，王耀钧，姜丽清 . 网络钓鱼防御技术研究 [J]. 信息网络安全，2012，(04)：30-35.

[9] 方欣，万扬，文霞等 . 基于协议分析技术的网络入侵检测系统中 DDoS 攻击的方法研究 [J]. 信息网络安全，2012，(04)：36-38.

[10] 范光远，辛阳 . 防火墙审计方案的分析与设计 [J]. 信息网络安全，2012，(03)：81-84.

[11] 梁宏，刘佳男，李勇 . “ 火焰 ” 病毒分析与防范 [J]. 信息网络安全，2012，(08)：157-159.

[12] 曹子建，赵宇峰，容晓峰 . 网络入侵检测与防火墙联动平台设计 [J]. 信息网络安全，2012，(09)：12-14.

[13] 杨青锦 . 一种信息安全、实现简单的移动设备位置信息交互方式 [J]. 中国科技信息，2013，(22)：111-113.

[14] 韩伟红，隋品波，贾焰 . 大规模网络安全态势分析与预测系统 YHSAS[J]. 信息网络安全，2012，(08)：11-14.

[15] 宋岩，王天然，徐皑冬 , 等 . 控制系统 Safe-Sec 安全通信方法研究 [J]. 自动化仪表，2013，34 (11)：30-33.

[16] 蒲石，陈周国，祝世雄 . 震网病毒分析与防范 [J]. 信息网络安全，2012，(02)：40-43.

[17] 龙丽萍，陈伟建，杨拥军，等 . 基于双因子认证技术的 RFID 认证协议的设计 [J]. 计算机工程与设计，2013，34 (11)：3726-3730.

[18] 余晓姿，马兆丰，钮心忻，杨义先 . 基于分类的未知病毒检测方法研究 [J]. 信息网络安全，2012，(11)：48-51.

[19] 乔宏明，姚文胜 . 基于策略提升公共云存储信息安全水平的方案研究 [J]. 移动通信，2013，(21)：53-57.

[20] 魏为民，袁仲雄 . 网络攻击与防御技术的研究与实践 [J]. 信息网络安全，2012，(12)：53-56.

[21] 杨继武 . 基于稀疏距离入侵特征表达的信息安全检测算法 [J]. 科技通报，2013，29 (10)：24-25.

[22] 朱峰 . 虚拟社会防控体系研究 [J]. 信息网络安全，2012，(08)：230-231.

[23] 张雪峰，周顺先 . 基于进程防火墙与虚拟盘的非法信息流过滤方法 [J]. 微型机与应用，2013，32 (20)：51-53.

[24] 王希忠，曲家兴，黄俊强，等 . 网络数据库安全检测与管理程序设计实现 [J]. 信息网络安全，2012，(02)：14-18.

[25] 李淳，赵建保，申晓留 . 多评估时间段的网络安全态势感知方法 [J]. 计算机应用，2013，33 (12)：3506-3510.

[26] 曲洁，朱建平 . 等级保护与关键基础设施防护的融合研究 [J]. 信息网络安全，2011，(12)：84-88.

[27] 赵少卡，李立耀，凌晓，等 . 基于 OpenStack 的清华云平台构建与调度方案设计 [J]. 计算机应用，2013，33 (12)：3335-3338.

[28] 戴瑾，刘波，卞皓宇 . 基于云计算的电子邮件安全服务系统的设计与实现 [J]. 计算机应用，2013，33 (12)：3350-3353.

[29] 吴轩亮 . 三网融合下城域网 DDoS 攻击的监测及防范技术研究 [J]. 信息网络安全，2012，(03)：45-48.

[30] 李春艳，张学杰 . 基于高性能计算的开源云平台性能评估 [J]. 计算机应用，2013，33 (12)：3580-3585.

[31] 黄俊强，方舟，王希忠 . 基于 Snort-wireless 的分布式入侵检测系统研究与设计 [J]. 信息网络安全，2012，(02)：23-26.

(C)1994-2021 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net