Implementation of Program Behavior Anomaly Detection and Protection Using Hook Technology

SHEN Jianfang[1] CHENG Lianglun[2] FU Xiufen[1]
1. Faculty of Computer, Guangdong University of Technology, Guangzhou, 510006; 2. Faculty of Automation, Guangdong University of Technology, Guangzhou, 510006

Abstract

Windows is an operating system based on message which is built on event—driven mechanism. Hook is one of surveillance point of message processing mechanism in Windows system. In this paper using Windows kernel technology, using Hook Service Table to replace Native's API, detect process and thread behavior, and realize detection and protection of registry and file and process. A program behavior anomaly detection and protection system is designed and implemented in Windows operating system. Hook and some key techniques of Hook are introduced, system frame and key technology of this system. At last, the experimental result validated the feasibility and availability of this system.

1. Introduction

With the development of Internet, there are many host computers in network. The question of computer system safety became more and more important. At present, most individual user and some small and medium-sized enterprises use Windows as their operating system. Hence program behavior anomaly detection and protection became more and more important in Windows operating system. In this paper, program behavior anomaly detection and protection are discussed, realize registry and file and process detection function.

The essence of program behavior anomaly detection and protection system is a system monitor .Its theory is checking operation object is protected or not when registry being written or deleted. If object is protected, then chose different scheme according to user decision, such as pass, access denied, terminate process or behavior trust. When an executable file is loaded, it must be written in internal memory. But this request can be caught at first time. It ensures run file is clean. And virus has no chance to run. Secondly all viruses must modify registry in order to auto run when machine is power on. And this request also can be caught in time. Thus further prevent virus outbreak. Thirdly virus may inject their thread into system process, and deceive firewall or antivirus software. This request can be caught and be refused. Therefore realization of detection and protection of registry and file and process can avoid infection from viruses and malicious program such as Trojan. Thus keep system Security.

2. Introduction to Hook and Related Technologies

2.1. Hook and Hook API technology introduction

Hook is one kind of system mechanism in Windows which replace interrupt mechanism in DOS. In fact, hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure. Everybody is familiar with translation software antivirus software and firewall etc. All are realized by Hook technology. In this paper, Hook API of Hook technology was used to realize system monitor, to avoid infection from viruses and malicious program such as Trojan.

Hook API is one kind of Hook function which replaced system API. As everyone know, API which provided by system need to be called in running programs. And Hook API is user-defined functions inserted before system API. All can jump to user-defined functions as long as program called this API. User can select original API function, and also can directly neglect to not dealing with it. To sum up, Hook like a barrier in street, all program will be examine before passed. It is important to note that function calling convention, function value type and

978-0-7695-3501-2/09 $25.00 © 2009 IEEE
DOI 10.1109/CMC.2009.225

function parameters of Hook must in complete accordance with original API function.

2.2. Realization and Application of Hook

According to sphere of action, Hook API can be divided into user Hook and system Hook [1]. User Hook is Win32 API, these functions usually all are overt, and can know their parameters, and easily realize relatively. But only can be used in given process, and can only limit in user mode and can not expand in kernel mode. System Hook is system service, namely Native API (this kind of API belongs to kernel API, usually become Win32 AP and used for user application) .Hook function work in kernel mode, must use the drive procedure to realize, it can be used in whole system. But the shortcoming is: many Native API functions are not overt, so being hard to definite its function parameter, and owing to work in kernel mode, need more stability and reliability, so difficult to realize. On various factors, the system in this paper adopts system Hook for more stability and reliability.

Realization and application of system Hook are as follow.

(1) Hook INT 2E method

This method is used to track or analyze system call. The principle is to replace INT 2E interrupt mechanism and let it direct user interrupt service routine.

(2) Hook PE method

This method is used to intercept or analyze another function call in kernel. The principle is to replace the corresponding function in PE export Table.

Some skills should be used in this method. There are no GetModuleHandl's ( ), GetProcAddress ( ) module etc directly to provide module address. User need to write like that, use a non-public function and a structure here. ZwQuerySystemInformation and SYSTEM_MODULE_INFORMATION to be used to get module base address. The function of PE export table can be replaced. But these educe another question again, that page attribute of kernel data is only read and can not be changed. There is no similar function like VirtualProtectEx ( ) of application layer to modify page attribute in kernel mode [2]. User need to write like that. User can revise write-protect of cr0 register, so just intercepting in kernel can be realized.

(3) Hook Service Table method

This kind of method is used to intercept Native API. The principle is that achieve intercept purpose by way to replace corresponding address of Native API in Service Table. Three are three detections, registry and file and process detections, all use Hook Service Table to replace Native API.

3. Program behavior anomaly detection and protection

3.1. System frame and Data flow

Windows CPU runs respectively in different mode. As is shown in figure 1, one is kernel mode, correspond to 80x86 ring0 layer, is kernel of operating system, all device drivers run in this mode. Another is user mode, correspond to 80x86 ring3 layer, is interface of operating system as well as all applications run in this mode [1].

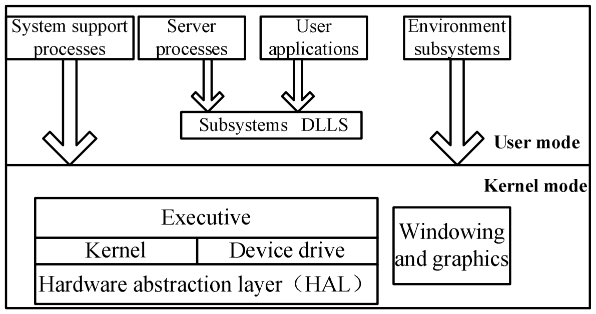


Figure1. Windows architecture

A program behavior anomaly detection and protection system is designed and implemented in Windows operating system.

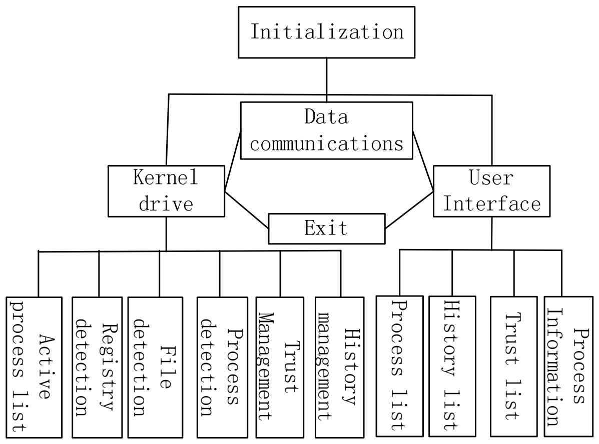


Figure2. System frame

System frame is shown in figure 2, adopt kernel mode in coordination with user mode, accomplished the detection in whole system by way of intercommunication between drive procedure and user procedure [4]. First layer data flow between user and procedure is shown in figure 3. Second layer data flow between user process and system process is shown in figure 4.

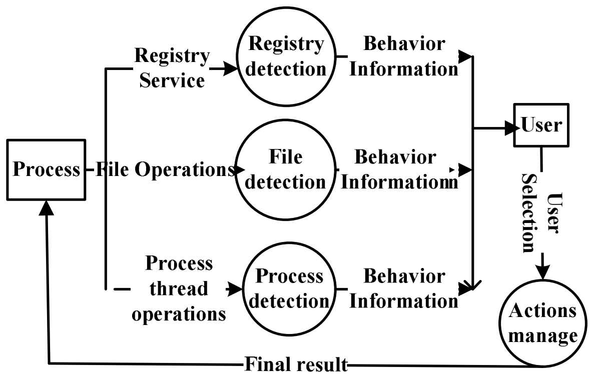


Figure3. **First layer data flow**

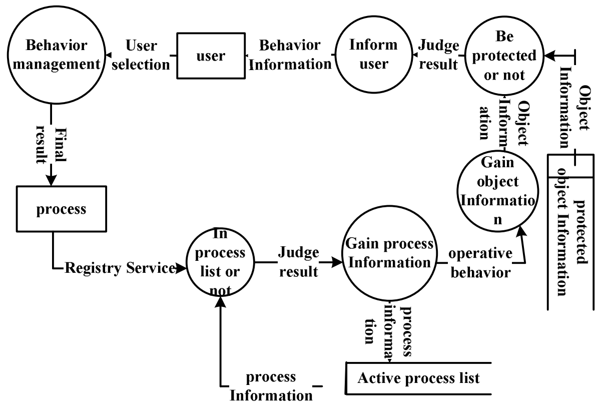


Figure4. **Second layer data flow**

4. Key technology of detection system

4.1. Call system service

User interface of system service (Native API) were provided by the form of wrapper function, and these wrapper functions are all in NTDLL.dll. These wrapper functions use INT 2E into kernel and call system service. Each system serve is marked by a service ID, and wrapper functions sent service ID of system service into EAX register, and sent the pointer of stack to the EDX register, then send INT 2E Instruction, INT 2E copy parameters from stack in user mode to stack in kernel mode [3]. This processing program of INT 2E is provided by ntosknrl.exe, is called KiSystemService. KiSystemService enters into internal structure KiServiceTable by way of the index values of EAX.

KiServiceTable can search and deal with entrance address of API. Before call objective function, KiSystemServie ( ) search about KiArgumentTable's structure, in order to search out the number of parameters which have been passed in stack, then according to this value copy the parameter to kernel stack. Hereafter, only need a simple assemble instruction named CALL carry out API procedure.

4.2. Call SDT (system dispatch table)

Two system services (SDT) exist in the system behind Win2000. They correspond to two different system services. These are KeServiceDescriptorTable and KeServiceDescriptorTableShadow respectively. There is only KeServiceDescriptorTableShadow in Windows NT. System services in ntoskrln.exe are defined in KeServiceDescriptorTable[3]. Its structure is shown in figure 5.

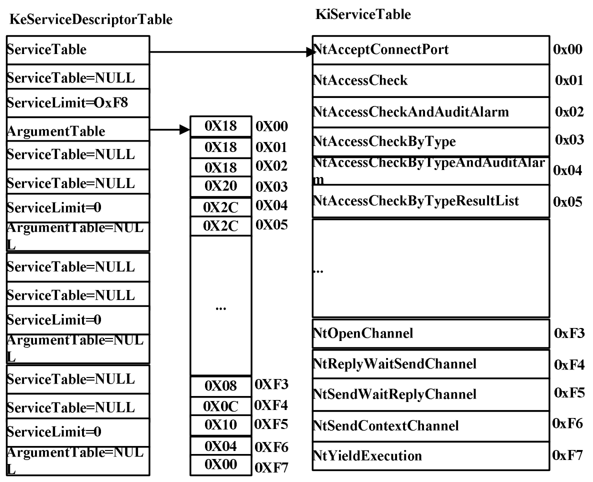


Figure5. **KeServiceDescriptorTable structure**

The interface functions which provided in kernel32.dll and advapi32.dll usually all is used KeServiceDescriptorTable. Winuser function and GDI function exist in win32k.sys at the same time. They are another kind of system service, corresponding table is KeServiceDescriptorTableShadow, provided the service USER and GDI in kernel mode. Service added by KeAddSystEmServiceTable can be copied to KeServiceDescriptorTable and KeServiceDescriptorTableShadow at the same time.

Data structure of SDT(system dispatch table) is shown follow.

```
typedef                                      struct
_SERVICE_DESCRIPTOR_TABLE       // SDT
  {SYSTEM_SERVICE_TABLE         ntoskrnl;
        // ntoskrnl.exe (Nativate API)
        SYSTEM_SERVICE_TABLE        win32k;
      // win32k.sys (gdi/user support)
        SYSTEM_SERVICE_TABLE        table3;
    // not used
        SYSTEM_SERVICE_TABLE        table4;
     // not used
  } SERVICE_DESCRIPTOR TABLE;
typedef struct _SYSTEM_SERVICE_TABLE
```

```
// SSD
{       PNTPROC    ServiceTable ;
  // array of entry points
       PULONG        CounterTable ;
   // array of usage counters.
       ULONG         ServiceLimit ;
    // number of table entries
       UCHAR*        ArgumentTable ;
   // array of bytes counts
} SYSTEM_SERVICE_TABLE
```

It is simple to visit KeServiceDescriptorTable, because it is openly led out by ntoskrnl.exe. Add " extern PSERVICE_DESCRIPTOR_TABLE KeServiceDescriptorTable", can visit data structure directly. But KeServiceDescriptorTableShadow does not be lead out (have no this table in NT), its place is different in Windows2000 and Windows XP, all Hook functions are led out by ntoskrnl.exe , so need not visit KeServiceDescriptorTableShadow.

4.3. Gain ID of Hook service

Services of Ntosknrl.exe are saved in SYSTEM_SERVICE_TABLE.ServiceTable. This member is linear array, and address pointers of service are saved in every item, as long as address pointer of function is replaced address pointer of provide function. How to know Hooked function ID (index)? The way of implement is follow.

```
mov eax, ServiceId
lea edx, ParameterTable
int 2eh
ret ParamTableBytes
```

ID of system service is in the 2nd byte place, so can get ID through macro.

```
#define SYSCALL(_function)  ServiceTable-
ServiceTable[ *(PULONG)((PUCHAR)_function+1)]
```

ServiceTable->ServiceTable is the number of ServiceTable in KeServiceDescriptorTable.

_ function+1 is address ServiceID.

4.4. Traverse ActiveProcessLinks

Every process in the Windows system has own EPROCESS [1]. Not only contains a lot of process information, and still a lot of pointers of related data structure. Such as every process ETHREAD as thread definition, process name at least, and PEB (Process Environment Block) in user space. Except PEB is in user space, other all are in system space. Main members of EPROCESS are shown in figure 6.

ZwQuerySystemInformation also finding the activity process list head to query process, then traverse ActiveProcessLinks. Basic information of every EPROCESS is returned (including process ID).

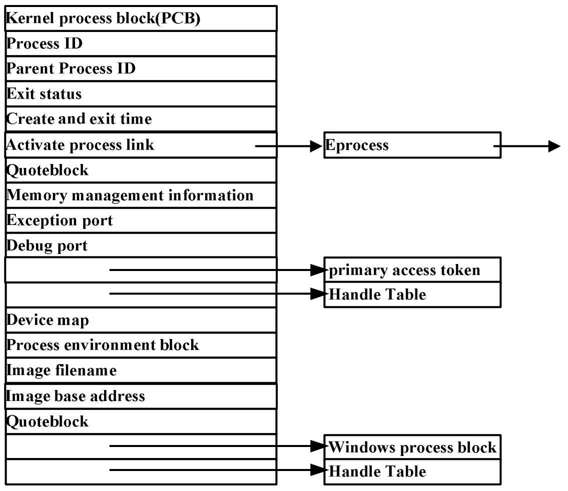


Figure6. EPROCESS Structure

The following procedure code is using EPROCESS to gain activity process ID.

```
ULONG GetProcessID()
{  ULONG         curproc;
   //get EPROCESS address of Current process
   curproc = (ULONG)PsGetCurrentProcess();
   if(curproc == (ULONG)0 || curproc ==
(ULONG)0xFFFFFFFF)
      return 0;
   // get ProcessId address in EPROCESS
   curproc += 0x84;
   if((curproc = *(ULONG*)curproc) == 0) return 0;
   curproc = (ULONG)PsGetCurrentProcessId();
   // return process ID
   return curproc;}
```

4.5. Intercommunication and data transmission

Intercommunication is difficulty and key technology in communication between drive procedure and user procedure. There is no ready-made API to realize intercommunication in Windows, usually adopt two ways to realize intercommunication.

(1) Asynchronous procedure call（APC）

An asynchronous procedure call (APC) is a function that executes asynchronously in the context of a particular thread. When an APC is queued to a thread, the system issues a software interrupt. The next time the thread is scheduled, it will run the APC function. An APC generated by the system is called a kernel-mode APC. An APC generated by an application is called a user-mode APC. A thread must be in an alert state to run a user-mode APC.

(2) Event object

Event object is an asynchronous object whose handle can be specified in one of the wait functions to coordinate the execution of multiple threads. More than one process can have a handle to the same synchronization object, making inter process synchronization possible. Created by the FindFirstChangeNotification function, its state is set to signaled when a specified type of change occurs within a specified directory or directory tree.

In this paper, use event object technology to realize intercommunication between drive procedure and user procedure.

There are two kinds of methods to realize data transmission.

(1) Shared memory object

To transfer data, multiple processes can use memory-mapped files that the system paged file stores. The first process creates the file mapping object by calling the CreateFileMapping function with INVALID_HANDLE_VALUE and a name for the object. By using the PAGE_READWRITE flag, the process has read/write permission to the memory through any file views that are created. Then the process uses the file mapping object handle that CreateFileMapping returns in a call MapViewOfFile to create a view of the file in the process address space. The MapView of file function returns a pointer to the file view, pBuf. The process then uses the CopyMemory function to write a string to the view that can be accessed by other processes.

(2) Shared memory address mapping

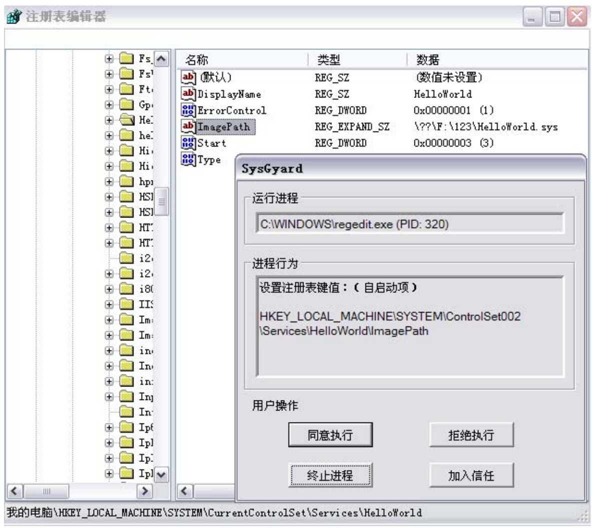
ExAllocatePoolXxx function allocates non-paged memory. Founds MDL to describe buffers, MmMapLockedPages function mapped memory address to user space.

The following procedure code is used to realize data transmission.

```
    case DAT_SHMEM_CRT:
    DbgPrint("Data Shared Memory Creat\n");
    pS_Buf= ExAllocatePoolWithTag( NonPagedPool,
    PAGE_SIZE+PathLen,
                 'DaSM');
      if(!pS_Buf)
       {  DbgPrint("Shared Memory Creat Error!\n");
       break; }
    pMdl     =     IoAllocateMdl(     pS_Buf,
PAGE_SIZE+PathLen,
               FALSE, FALSE, NULL);
 if(!pMdl)
    {  DbgPrint("MDL Creat Error!\n");
       ExFreePool(pS_Buf);
       break;       }
```

5. System Verification

In this paper, take registry detection is as example. Startup, hosts File, internet settings, internet explorer, toolbar, policy of registry are easy to attack. Regedit modify these six items directly. The test result shows in figure 7, and validate the feasibility and availability of system.



名称	类型	数据
(默认)	REG_SZ	(数值未设置)
DisplayName	REG_SZ	HelloWorld
ErrorControl	REG_DWORD	0x00000001 (1)
ImagePath	REG_EXPAND_SZ	\??\F:\123\HelloWorld.sys
Start	REG_DWORD	0x00000003 (3)
Type		

C:\WINDOWS\regedit.exe (PID: 320)

设置注册表键值：(自启动项)

HKEY_LOCAL_MACHINE\SYSTEM\ControlSet002
\Services\HelloWorld\ImagePath

我的电脑\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\HelloWorld

Figure7. Test pattern of registry detection

References

[1] David A. Solomon, Mark E. Russinovich, Microsoft Windows Internals, 4thed, Microsoft Press, 2004

[2] Chu Kuangren, Developing Applications with file system, 2thed, Shanghai, 2007

[3] Jefrey Richter, Programming applications for Microsoft Windows, 4thed, Beijing, Engineering Industry Publishing House, 2004：546-547

[4] HAO Dongbai, GUO Lin, HUANG Hao, Design and implementation of program behavior anomaly detection system based on system service hook, Compeer Engineering and Design, Beijing, 2007:4373-4376

Foundation Item: This project is supported by Natural Science Foundation of Guangdong Province [No.06021484] and Technical and Scientific Project of Guangdong Province [No.06A1104412]

Biography: Shen Jianfang, Female, Lecturer, PhD candidate, Guangdong University of Technology, Main research directions is intrusion detection and multimedia technology, E-mail: tysjf@gdut.edu.cn