Detecting keylogger virus by monitoring keyboard driver stack

Asst.Lecturer Farah Majid*

1- Abstarct

This work is devoted to design and implement a software to monitor keyboard driver stack for any illegal embedding of malicious filter driver. Filter drivers is the effective tool used by keylogger software to record user keystrokes. Recording keystrokes is a very hostile action and it is mostly done by viruses.

Enumerating the size of the drivers stack dedicated for the keyboard device and the location of upper most filter driver. A filter driver is designed along this paper using Microsoft Driver Development Kit (DDK) 2003, this filter driver is going to be attached to the keyboard driver stack to be the upper most keyboard filter driver. Another user level program is designed to interact with the filter driver. When Windows I/O manager will send Input/Output Request Packet (IRP) the filter driver will intercept that packet and send back to the user level program specially designed along this paper. The stack depth and stack location will be retrieved from IRP sent by the filter driver.

* Ministry of Higher Education & Scientific Research (SCR office)

1- Introduction

Keystroke logging (often called keylogging) is the action of tracking (or logging) the keys struck on a keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored. There are numerous keylogging methods, ranging from hardware and software-based approaches to electromagnetic and acoustic analysis [1] .

A keylogger builds a log of everything typed into a keyboard to be reviewed by a third party. Keyloggers can be used for legitimate purposes to troubleshoot networks, analyze employee productivity, or to assist law enforcement, for example; or they can be used for illegitimate purposes to surreptitiously spy on people for personal gain. A keylogger can be a hardware device or a software program [2] .

Software keyloggers are often installed through malware like Trojans, viruses, spyware or rootkits. These keyloggers can collect keystrokes through a number of methods, depending on design. Some keyloggers work at the kernel level; others use a hook to hijack system processes that manipulate the keylogger; and still others use entirely different means. A keylogger that is installed remotely through malicious means secretly sends its logs to the person who planted the device via an Internet connection[2] .

The threat of the keylogger is rapidally increasing and become devastating when it comes to the most potential threat facing the whole world , which is the (Botne). This viruse hit more 60% percent of the World Wide Web and causing damages bigger than it was origionally designed for. This paper is not involved in explaining how 'Botnet' works, all it concerns here that

‘Botnet’ is using keylogger as a main tool to compromise host security that deploys encryption software to protect ‘Botnet’ from harvesting real information, ‘Botnet’ is using keylogger to get information entered by the user before going to be encrypted [2] .

More than 25 millions computers are infected with keylogge as an estimation all over the World Wide Web. Although, many techniques are used to prevent keylogger, it is still threating hosts all over the world.

This paper will introduce a different approach that combine plug in hardware device (USB flash disk) and kerenl level software driver component .

2- Keylogger types

Key stroke logging (often called Keylogging) is the action of tracking (or logging) the keys struck on keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored . [1,2]

3-1 Software keylogger

Software keyloggers are which are often in installed through other software carriers called malware like Trojans, viruses, spyware or rootkits. These keyloggers can collect keystrokes through a number of methods, depending on design. Some keyloggers work at the kernal level; others use hook to hijack system processes that manipulate the keylogger; and still others use entirely different means. A keylogger that is installed remotely through malicious means secretly sends its logs to the person who plantes the device via an internet connection. [1,2]

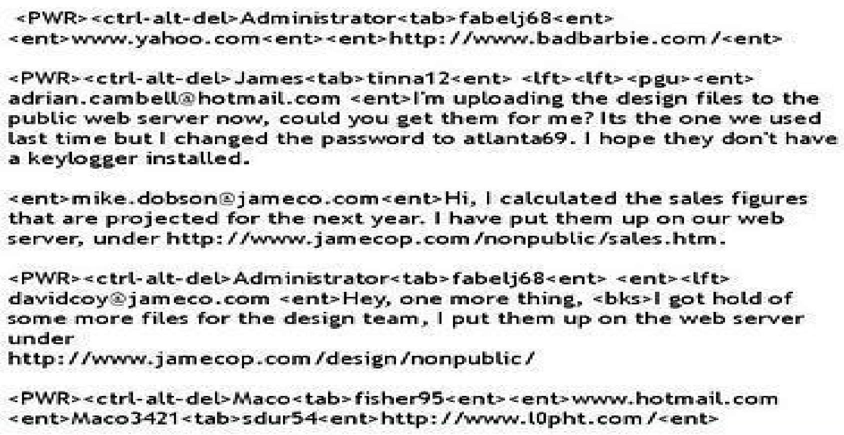
3-1-1 Classic keylogger

Software is the dominant method of monitoring computer activity. It has advantages that it can transmit activities over a network, and may be very affordable. Many different software programs have been created to address the need to monitor (over 300). In response, there is software which is designed to detect or disable the monitoring software. Keystroke recording software has existed almost since the arrival of the first computers. These programs create a log of all keystrokes typed and store the log file on the computer hard drive. These programs are generally interrupt-driven (from the keyboard interrupt). Thus, it consumes computer time while it reads the keystrokes and writes them to the computer hard drive. Further, the file on the hard drive may be discovered and erased/modified.

3- 1-2 Modern keylogger

Modern software keystroke recorders have evolved beyond simple key loggers. Some will record the screen images, and play them back like a VCR. Other programs will e-mail the keystroke logs to a remote computer. Nonetheless, these programs still reside on the hard drive (where they can be detected) and also consume computer time. Storing screen shots to your hard drive is disastrous for computer performance, loading down the CPU, RAM, and hard drive. Storing screen images on the hard drive is like buying a high-performance boat and cruising around with the anchor down.

Many software monitors have come under the gun for their tendency to make the system unstable.



```
<PWR><ctrl-alt-del>Administrator<tab>fabelj68<ent>
<ent>www.yahoo.com<ent><ent>http://www.badbarbie.com/<ent>

<PWR><ctrl-alt-del>James<tab>tinna12<ent> <lft><lft><pgu><ent>
adrian.cambell@hotmail.com <ent>I'm uploading the design files to the
public web server now, could you get them for me? Its the one we used
last time but I changed the password to atlanta69. I hope they don't have
a keylogger installed.

<ent>mike.dobson@jameco.com<ent>Hi, I calculated the sales figures
that are projected for the next year. I have put them up on our web
server, under http://www.jamecop.com/nonpublic/sales.htm.

<PWR><ctrl-alt-del>Administrator<tab>fabelj68<ent> <ent><lft>
davidcoy@jameco.com <ent>Hey, one more thing, <bks>I got hold of
some more files for the design team, I put them up on the web server
under
http://www.jamecop.com/design/nonpublic/

<PWR><ctrl-alt-del>Maco<tab>fisher95<ent><ent>www.hotmail.com
<ent>Maco3421<tab>sdur54<ent>http://www.l0pht.com/<ent>
```

Figure 1: Sample text recorded on infected host with keylogger

3-2 Hardware keylogger

Hardware keystroke recorders contain two main components: a simple microprocessor, and non-volatile memory. The microprocessor handles tasks such as: interpreting keystrokes, checking for the access password, and displaying menu options. The non-volatile memory is a fairly large memory, which is used to store the keystrokes. Non-volatile memory retains data even during a power loss. This allows a hardware keystroke recorder to be unplugged and still retain the keystroke log. Further, the ability to retain the keystroke log even when unplugged, makes it a portable device. It can be used to record on one computer, and can be read out on another computer.

4-Keyboard Driver Under Windows environment

The keyboard is a device connected to the motherboard through a hardware controller. This controller basiclly intiated by motherboard BIOS (Basic Input/Output System), but when installing an operating system a new driver (or drivers) is installed and steer out the basic behaivor to cope with the environment imposed by the operating system (e.g., windows operating system dose not allow the user software to directly contact the keyboard). The following is the Device drivers are running in the most priviledged level of the system, which is ring 0 level, user mode software is running in ring 3, which is the least privilegded level as shown in Figure (2). [1,3]

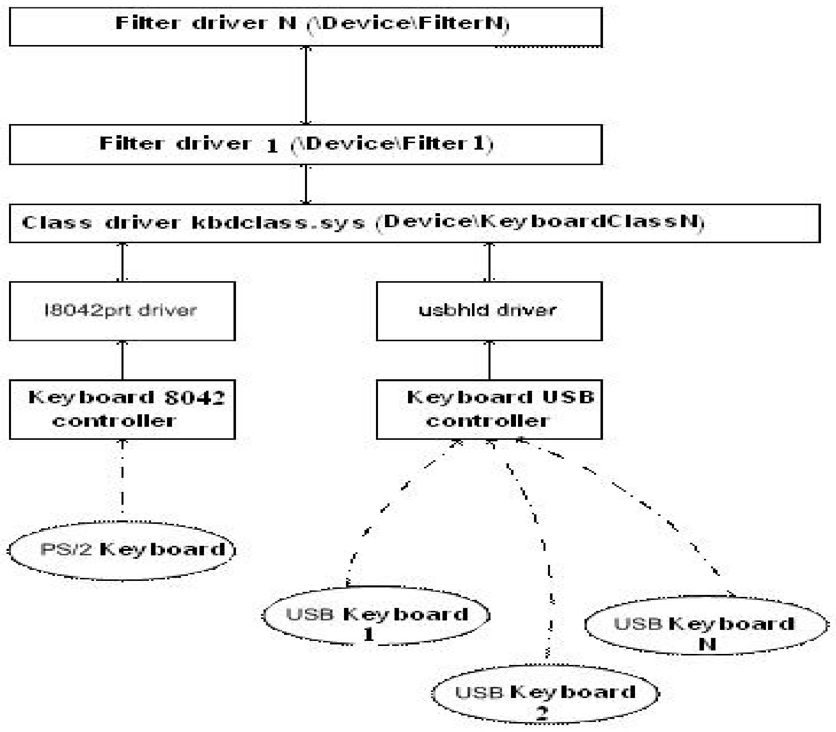


Figure 2: Shows the system architecture of keyboard device and the driver

5-Input/Output Request Packet and Driver Stack

Input/Output Request Packet (IRP) is a data structure created by I/O Manager to communicate kernel level drivers. The first to accept IRP is the highest driver in the stack, and correspondingly the last one to get it is the driver responsible for the interaction with the real device. By the moment of IRP creation, the number of drivers in the stack is known. I/O Manager allocates some space in IRP for IO_STACK_LOCATION structure for each driver. Also the index and pointer of the current IO_STACK_LOCATION structure are stored in the IRP header.[1,3,4]

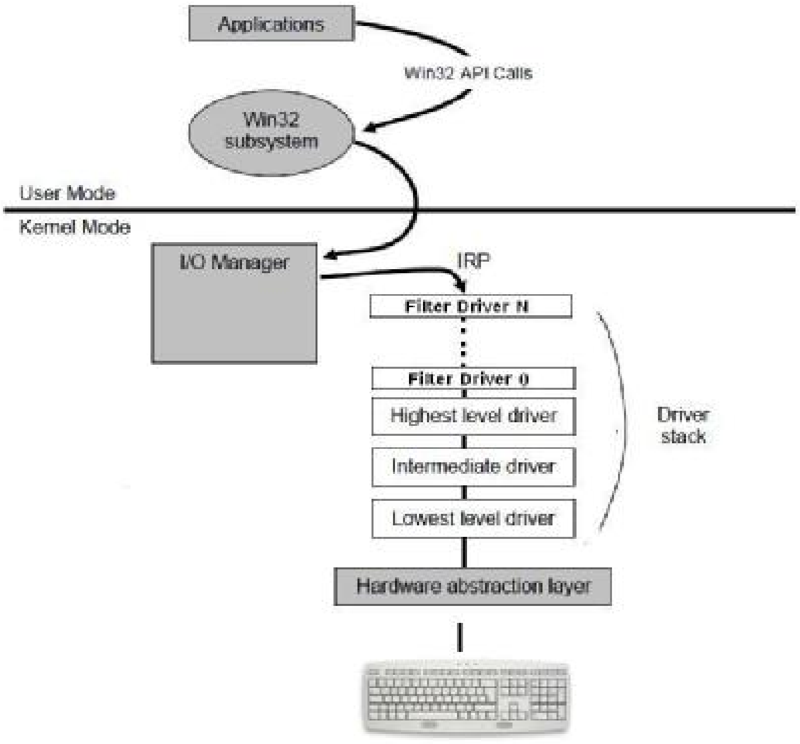


Figure 3: shows the communication scheme between the application and the hardware

The drivers form the chain with IRP as the data medium. Correspondingly the simplest way to hook data from the device driver (and keyboard driver in particular) is to attach own specially developed driver to the stack with the existing ones [1,3].
Several *Stack Locations* might exist within an IRP. One for each device belonging to the chain of layers that follow until the IRP reaches the target driver. In other words, there is a Stack Location for the target driver and an additional one for each filter driver installed on it.[1,3,4]

```
typedef struct _IRP
{
    SHORT Type;
    WORD Size;
    PMDL MdlAddress;
    ULONG Flags;
    ULONG AssociatedIrp;
    LIST_ENTRY ThreadListEntry;
    IO_STATUS_BLOCK IoStatus;
    CHAR RequestorMode;
    UCHAR PendingReturned;
    CHAR StackCount;
    CHAR CurrentLocation;
    UCHAR Cancel;
    UCHAR CancelIrql;
    CHAR ApcEnvironment;
    UCHAR AllocationFlags;
    PIO_STATUS_BLOCK UserIosb;
    PKEVENT UserEvent;
    UINT64 Overlay;
    PVOID CancelRoutine;
    PVOID UserBuffer;
    ULONG Tail;
} IRP, *PIRP;
```

Figure 4 : shows IRP (Input/Output Request Packet) structure

6-Monitoring keyboard driver stack

This paper is presenting the idea of locking the driver stack which prevent the installation of any new driver without get noticed. This is going to be done by enumerating the size of the keyboard driver stack and save it with other driver stack information in a flash disk. Any software that will try to log keyboard strokes has to add a filter driver within the driver stack which eventually will change the size of the stack.

The work presented by this paper is going to start by creating a upper level filter driver and attach it to keyboard driver stack as the following code shows:

```
PDEVICE_OBJECT pKeyboardDeviceObject = NULL;
   NTSTATUS lStatus = IoCreateDevice(pDriverObject, 0,NULL,
FILE_DEVICE_KEYBOARD   0,FALSE,  &pKeyboardDeviceObject);
```

After creating the device object it has to be attached to the keyboard driver stack as the following code shows:

```
m_pNextDevice = IoAttachDeviceToDeviceStack(m_pKBFilterDevice,
                                                          m_pNextDevice);

if (m_pNextDevice == NULL)
{    throw(runtime_error("an error has occurred while attaching to stack"));
                  }

         m_bIsAttached = true;);
```

The responsibility of this keyboard filter driver is to pass the IRP back to a custom application designed specifically to interpret IRP's fields. The most important fields are StackCount and StackLocation which shows the number of drivers in the drivers stack and the location of each driver within the stack.

```
PIO_STACK_LOCATION m_StackLocation = IoGetNextIrpStackLocation(MyIrp);
m_StackLocation->MajorFunction = IRP_MJ_INTERNAL_DEVICE_CONTROL;
m_StackLocation->Parameters.DeviceIoControl.IoControlCode =
                        IOCTL_CREATE_NEW_RESOURCE_CONTEXT;
```

Microsoft recommends that drivers register the device interface in order to send I/O control requests from an application to the driver. The keyboard filter driver designed and implemented by this paper will follow that recommendation and register itself as the following code shows .

```
retStatus = IoRegisterDeviceInterface (  KeyboardPhysicalDevice,
                                          (LPGUID) &KEYBORD_INTERFACE,
                                           NULL,     &devExt->GetIrp);
```

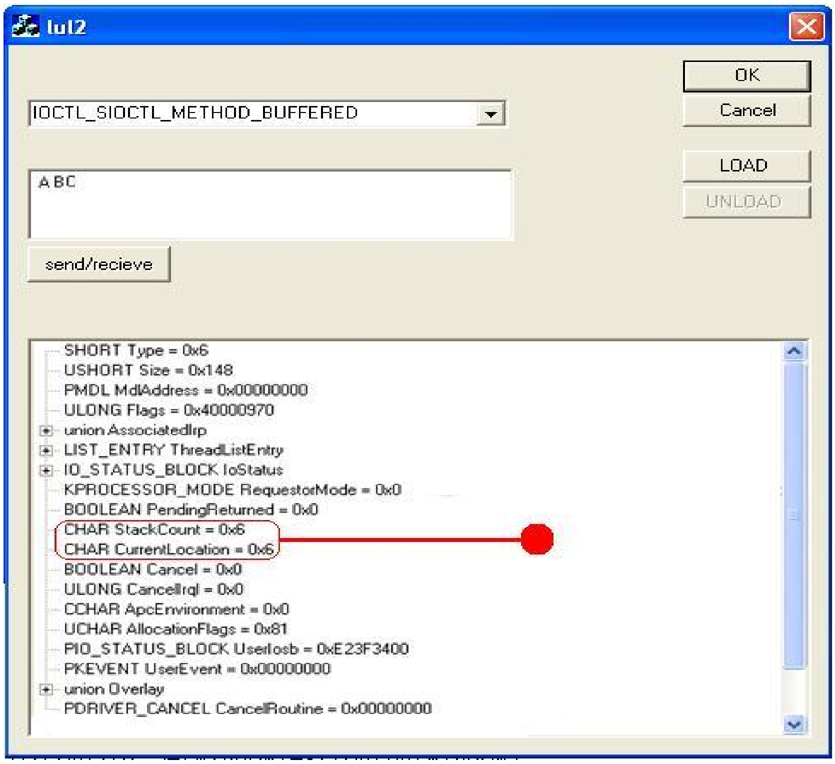
Now, the filter driver is ready to process I/O control request from application level. An application has been designed to send I/O request of the type METHOD_BUFFERED, the expected response from the filter driver is to send back all IRP information . by retrieving IRP information it will easy to read StackSize and StackLocation. the following code shows how filter driver sends back IRP information.

```
. PIO_STACK_LOCATION  irpSp = IoGetCurrentIrpStackLocation(
Irp );
   inBufLength = irpSp-
>Parameters.DeviceIoControl.InputBufferLength;
   outBufLength = irpSp-
>Parameters.DeviceIoControl.OutputBufferLength;
switch ( irpSp->Parameters.DeviceIoControl.IoControlCode )
   {
   case IOCTL_SIOCTL_METHOD_BUFFERED:
          SIOCTL_KDPRINT(("Called
         IOCTL_SIOCTL_METHOD_BUFFERED\n"));
```

```
PrintIrpInfo(Irp);
       inBuf = Irp->AssociatedIrp.SystemBuffer;
      outBuf = Irp->AssociatedIrp.SystemBuffer;
       outputLength = CopyIrpInfo(outBuf,Irp);
       Irp->IoStatus.Information = outputLength;
       Break;
```

.

The following screen shot shows the IRP information retrieved from upper most filter driver. It is obviously seen the important fields (StackCount = 0x6) and (CurrentLocation = 0x6), the values should be equal if the filter driver is the most upper class filter. In the following screen shot three strokes were happened and its corresponding IRP is collected .

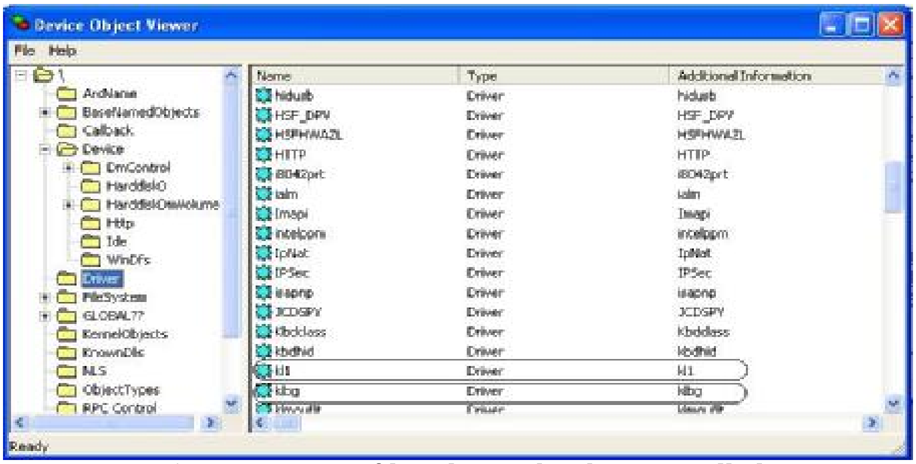


IOCTL_SIOCTL_METHOD_BUFFERED

ABC

```
SHORT Type = 0x6
USHORT Size = 0x148
PMDL MdlAddress = 0x00000000
ULONG Flags = 0x40000970
union AssociatedIrp
LIST_ENTRY ThreadListEntry
IO_STATUS_BLOCK IoStatus
KPROCESSOR_MODE RequestorMode = 0x0
BOOLEAN PendingReturned = 0x0
CHAR StackCount = 0x6
CHAR CurrentLocation = 0x6
BOOLEAN Cancel = 0x0
ULONG CancelIrql = 0x0
CCHAR ApcEnvironment = 0x0
UCHAR AllocationFlags = 0x81
PIO_STATUS_BLOCK UserIosb = 0xE23F3400
PKEVENT UserEvent = 0x00000000
union Overlay
PDRIVER_CANCEL CancelRoutine = 0x00000000
```

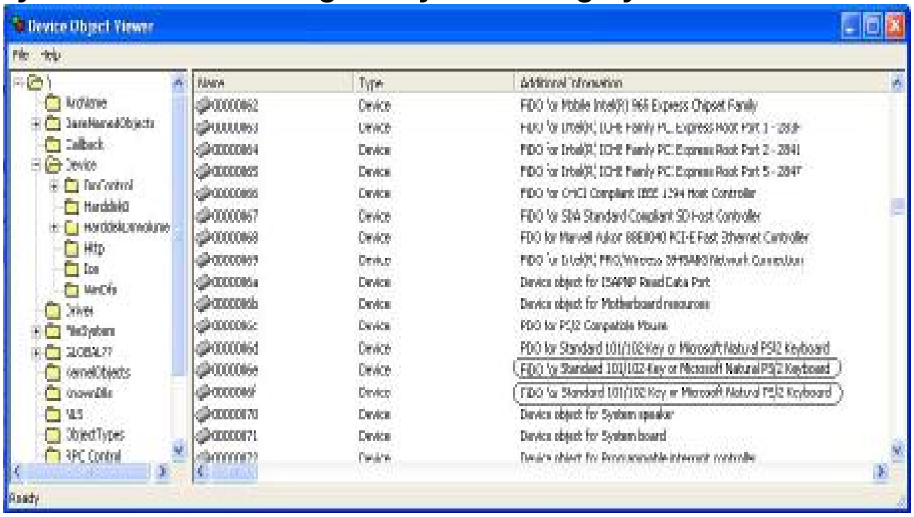
Figure 5 : Shows a screen shot of Keyboard IRP Tracker



Name	Type	Additional Information
hidusb	Driver	hidusb
HSF_DPV	Driver	HSF_DPV
HSFHWAZL	Driver	HSFHWAZL
HTTP	Driver	HTTP
i8042prt	Driver	i8042prt
ialm	Driver	ialm
Imapi	Driver	Imapi
intelppm	Driver	intelppm
IpNat	Driver	IpNat
IPSec	Driver	IPSec
isapnp	Driver	isapnp
JCDSPY	Driver	JCDSPY
Kbdclass	Driver	Kbdclass
kbdhid	Driver	kbdhid
kl1	Driver	kl1
klbg	Driver	klbg
[illegible]	Driver	[illegible]

Figure 6 : Two filter drivers has been installed

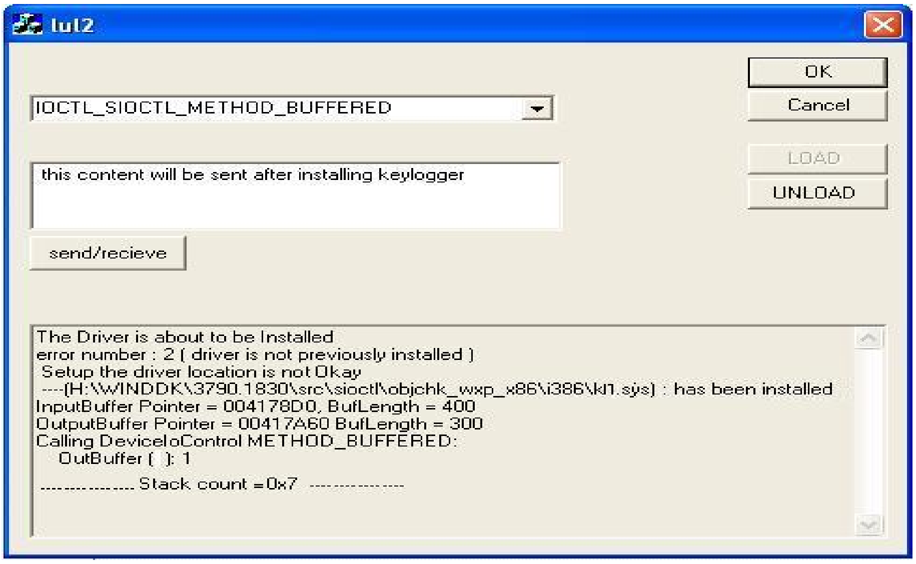
Figure (6) shows that two filter drivers has been installed one is kl1.sys which is the upper most filter driver that designed to serve the proposal of this paper, and klbg.sys which will act as third party keylogger. By installing these two filter drivers; two objects are to be added to the system devices, figure (7) shows the installed devices within the system after installing kl1.sys and klbg.sys filter drivers.



Name	Type	Additional Information
00000062	Device	FDO for Mobile Intel(R) 965 Express Chipset Family
00000063	Device	FDO for Intel(R) ICH8 Family PCI Express Root Port 1 - 283F
00000064	Device	FDO for Intel(R) ICH8 Family PCI Express Root Port 2 - 2841
00000065	Device	FDO for Intel(R) ICH8 Family PCI Express Root Port 5 - 2847
00000066	Device	FDO for OHCI Compliant IEEE 1394 Host Controller
00000067	Device	FDO for SDA Standard Compliant SD Host Controller
00000068	Device	FDO for Marvell Yukon 88E8040 PCI-E Fast Ethernet Controller
00000069	Device	FDO for Intel(R) PRO/Wireless 3945ABG Network Connection
0000006a	Device	Device object for ISAPNP Read Data Port
0000006b	Device	Device object for Motherboard resources
0000006c	Device	PDO for PS/2 Compatible Mouse
0000006d	Device	PDO for Standard 101/102-Key or Microsoft Natural PS/2 Keyboard
0000006e	Device	FiDO for Standard 101/102-Key or Microsoft Natural PS/2 Keyboard
0000006f	Device	FiDO for Standard 101/102-Key or Microsoft Natural PS/2 Keyboard
00000070	Device	Device object for System speaker
00000071	Device	Device object for System board
00000072	Device	Device object for Programmable interrupt controller

Figure 7 : shows two filter object has been created in keyboard device stack

Figure (8) shows that the system designed by this paper will detect the changes in keyboard device stack and rising an alert to user that a possible malicious driver has been inserted into the device stack.



```
this content will be sent after installing keylogger
```

```
The Driver is about to be Installed
error number : 2 ( driver is not previously installed )
 Setup the driver location is not Okay
 ----(H:\WINDDK\3790.1830\src\sioctl\objchk_wxp_x86\i386\kl1.sys) : has been installed
InputBuffer Pointer = 004178D0, BufLength = 400
OutputBuffer Pointer = 00417A60 BufLength = 300
Calling DeviceIoControl METHOD_BUFFERED:
    OutBuffer (  ): 1
 ................ Stack count =0x7  ................
```

Figure 8 : shows how the system detect the changes in device stack

Conclusions and Future Work

1. It is possible to secure driver stack and prevent embedding more drivers without notifying the I/O manager. One challenge is still a milestone which the possibility of hooking I/O manager modules itself, this could compromise the whole security of all I/O object management not just the keyboard driver but all I/O devices.

2. Windows layered architecture is a very powerful feature in the point of developing and modularization but it might be a security breech without an a robust athuentication protocols before adding new drivers to the stack.

3. It was proven during the examination of the code that it is possible to conduct DoS attack against computer devices by embedding filter driver with IoCompletion routine.

References

1- Art Baker and Jerry Lozano, "**The Windows 2000 Device Driver Book"**, second edition , prentice-Hall, 2001

2- Andrew S. Tanenbaum, "**Modern Operating Systems**", second edition , prentice-Hall 2001.

3- Microsoft Division, "**Microsoft Windows 2000 Driver Design Guide"**, Microsoft press,2000

4- Walter Oney, "**Programming the Microsoft Windows Driver Moderl**", second edition ,2003

5- http://www.codeproject.com/KB/system/driverdev4asp.aspx , " A simple demo for WDM Driver development, 2004

6- http://support.microsoft.com/kb/262305 , "How to send IOCTLs to a Filter Driver", 2005.

7- http://www.codeproject.com/KB/system/CServiceWrapper.aspx , "A C++ class wrapper to load/unload device drivers", 2008

الكشف عن فيروس كلوغر بواسطة تعريف مكدس لوحة المفاتيح

م.م. فرح ماجد*

هذا العمل مكرس لتصميم وتنفيذ برنامج مراقبة لمكدس سواقة لوحة المفاتيح لاستكمال الحشر الغير قانوني لسواقات أخرى ضمن هذا المكدس . إن حشر سواقات مرشحة في مكدس لوحة المفاتيح يعتبر من الادوات الفعالة التي تستخدم في برمجيات التجسس من نوع مسجلات المفاتيح والتي تقوم بتسجيل جميع ضربات المستخدم على لوحة المفاتيح . يصنف هذا التصرف على أنه عدائي جدا وذلك لامكانية استلال المعلومات السرية الخاصة بالمستخدم .

ان هذا البحث سوف يبرهن على ان حساب حجم المكدس المخصص للوحة المفاتيح وتسلسل السواقات المرشحة يمكن ان يستخدم بفعالية عالية لاستمكان الاختراقات حيث ان البحث سوف يطرح تصميم سواقة خاصة من نوع المرشحات والتي سوف يتم حشرها لتكون في قمة المكدس على الدوام . ان هذه السواقة سوف يتم برمجتها لتعطي معلومات عن عمق المكدس وتسلسل وجودها في المكدس . سوف يتم بناء السواقة بأستخدام الحقيبة البرمجية الخاصة من شركة مايكروسوفت (DDK) .

ان عمل السواقة المرشحة التي يطرحها هذا البحث ستكون مهمتها استكمال كل كمات البيانات المتبادلة (IRP) بين التطبيقات ولوحة المفاتيح والتي سوف يرسلها مدير بيئة النوافذ الى المكدس الخاص بلوحة المفاتيح ومن ثم استخراج عمق المكدس وتسلسل السواقة منها .

* وزارة التعليم العالي والبحث العلمي / دائرة البعثات والعلاقات الثقافية