Complete Security Package for USB Thumb Drive

Sinan Adnan Diwan [1,4], .Dr.Sundresan Perumal [2], Ammar J.Fatah [3*]

1. Wasit University,College of Computer and Mathematics,Computer Science dept. Wasit,Republic of Iraq
2. Limkokwing University of creative technology,information technology department ,cyberjaya,Selangor,Malaysia
3.Science Gate,Virtual Research Center ,System security dept.
4. Limkokwing University of Creative Technology,Information Technology Department, Cyberjaya,Selangor,Malaysia

Abstract

This paper is devoted to design and implement a complete security platform for USB flash disks due to the popularity of this device in exchanging data, it is a complete system security solution as it concerns the thumb drive due to the manipulation of I/O operation not the file system. USB flash disks have been the major threat for computer system beside the internet threats where viruses can spread from computer to computer or from computer to network.

USB complete security system presented by this paper is composed of three essential elements: kernel filter driver which will be installed in USB device driver stack to intercept all exchanged packets and send it to encryption unit, kernel level encryption/decryption unit and configuration unit.

In contrary to most USB security modules the system presented by this paper will store only the round number of the key generator with the encrypted data. Round number will be coded using MD5 algorithm to increase the immunity of attacking data stored in the flash disks.

Keywords: USB protection, kernel driver, device stack, encryption/decryption, filter driver, MD5.

1. Introduction

USB memory sticks can be found almost everywhere. Today, they can be seen as the replacement for floppy-disks, ZIP-drives and all that kind of media. A USB device is indeed a useful, economical way to transfer data. In fact, according to Gartner IT research and advisory company, there were roughly 222 million USB devices shipped in 2009. However, a recent study shows that though USB devices are a convenient means of transferring information, they can also serve as channels to transmit potential threats.[1]

Sunnyvale (California, US) based security firm, Narus Inc. recently listed the 10 "cyber threat trends" for 2011 and beyond, claiming that as attacks through USB becomes inexpensive and data is circulated through them at various public and private locations, the probability of computer Trojans and other malicious software is increasing.[2]

According to Panda’s report, 25% of worm based malware spreads through the USB drives. Even more, most of viruses are designed to spread via USB drives. Security Company confirms that cybercriminals are very persistent and put a lot of efforts to make user’s life impossible.[3]

25% of latest created malware are configured to enter the system through portable storage devices, usually USB drives.

Luis Corrons, the technical director of PandaLabs, the research arm of Panda Security comments: „Much of the malware in circulation has been designed to distribute through these devices. Not only does it copy itself to these gadgets, but it also runs automatically when a USB device is connected to a computer, infecting the system practically transparently to the user.“ [2]

A recent Panda survey has shown that 27% attacks of worm based malware discovered last year used USB hardware to get in. Corrons warns about the real threat to smartphones, cameras and music players. “All these devices have memory cards or internal memories and therefore it is very easy for a cell phone to be carrying a virus.[4,3]

Kaspersky Lab warns of a new type of malicious program, which currently gives up virus experts still riddle the Stuxnet Trojan. Alex Gostev, Virenanalyst bei Kaspersky Lab, nimmt in drei Blogbeiträgen die neue Windows-Zero-Day-Lücke genauer unter die Lupe. Alex Gostev, virus analyst at Kaspersky Lab, takes in three blog posts, the new Windows zero-day vulnerability more closely. [5]

Der Trojaner Stuxnet, von Kaspersky Lab als Trojan-Dropper.Win32.Stuxnet identifiziert, infiziert USB-Sticks mittels Lnk-Dateien und verbreitet sich anschließend durch die mobilen Datenträger über die Autorun-Funktion von Windows. The Trojan Stuxnet, Kaspersky Lab as Trojan-Dropper.Win32.Stuxnet identified from infected USB sticks by Link files and then spread through the mobile devices through the auto-run feature of Windows.[7]

2. Windows Architecture and Device Stack

Windows NT allows several driver layers to exist between an application and a piece of hardware. Thus drivers are grouped together in stacks that work together to completely process a request targeted at a particular device object.[5] The operating system uses a data structure known as an I/O Request Packet (IRP), to communicate with a kernel-mode device driver as it is presented in figure (1).[5]

Windows NT uses a layered driver model, starting in Bus driver tile it ends in upper filter driver, to process I/O Requests Packet. In this model, drivers are organized into stacks. Each driver in a stack is responsible for processing the part of the request that it can handle, if any. If the request cannot be completed, information for the lower driver in the stack is set up and the request is passed along to that driver.[2,5]

This layered driver model allows functionality to be dynamically added to a driver stack. It also allows each driver to specialize in a particular type of function and decouples it from having to know about other drivers.[2,5]

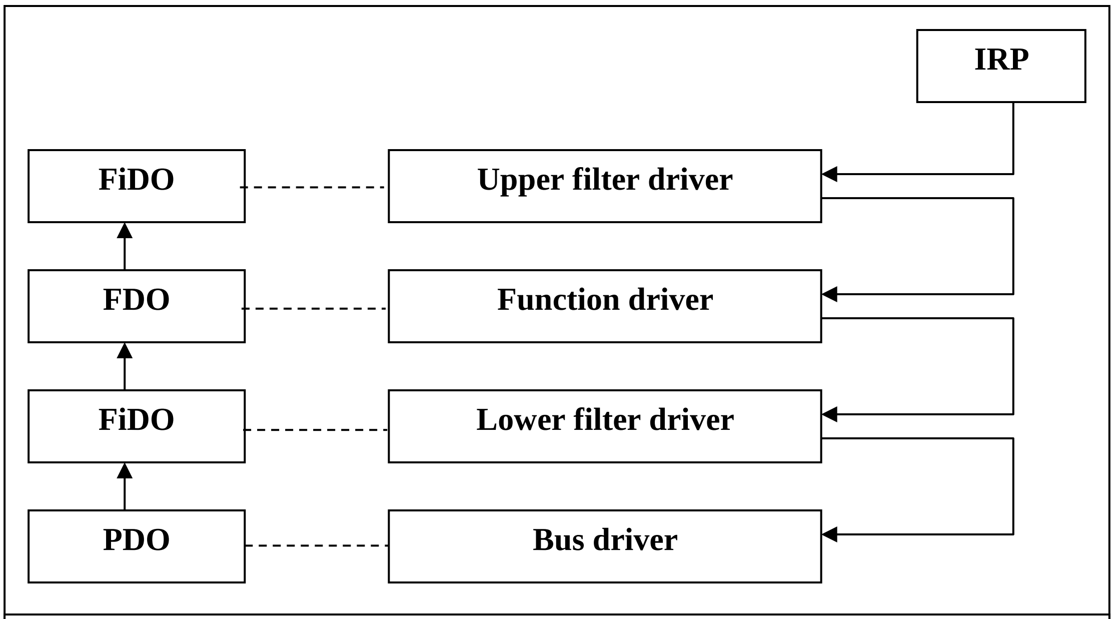


Figure 1 : Layering of device objects & drivers in the Windows Driver Model

In the windows Driver Model, each hardware device has at least two device drivers. One of these drivers, which is the function driver (FDO), is which it appears to be thought as the device driver; it understands all the details about how to make the hardware work. It's responsible for initiating I/O operations, for handling the interrupts what occur when those operations finish, and for providing a way for the end user to exercise any control over the device that might be appropriate. [1,4,5]

2.1 Functional driver

Function driver, this type of drivers can be either class or mini drivers, but they always act as an interface between abstract I/O request and the low-level physical driver code, function driver knows how to interpret all requested operations from the application level and upper drivers in the device stack, and without this driver the physical device can't even operate normally. [5,6,7]

2.2 Filter Drivers

A Filter Driver is a special type of layered driver. What sets a filter driver apart from the layered driver is that it is invisible. They attach themselves to any other driver and intercept requests directed at the lower driver's Device objects. It is developed primarily to allow the addition of new functionality beyond what is currently available. The filter driver may either use the services of the original target of the I/O request, or use the services of other kernel-mode drivers to provide value-added functionality.[4,5,8]

Filter drivers are divided into two classes, upper and lower class filters, what is above the function driver is upper driver while what is below the function driver is lower driver. Upper filter drivers see IRPs before the function driver, and they have the chance to support additional features that the function driver doesn't know about. Lower filter drivers see IRPs that the function driver is trying to send to the bus driver. [5]

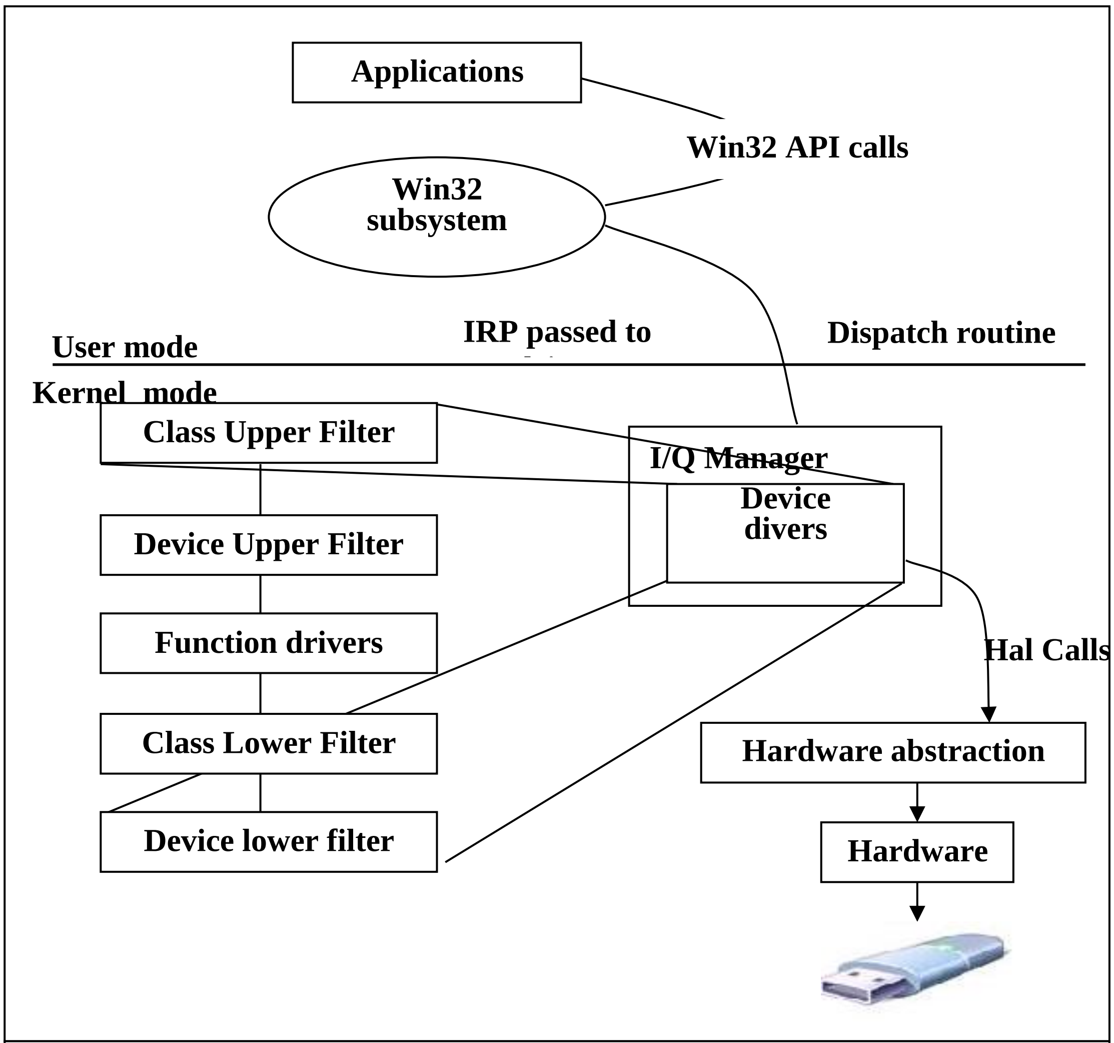
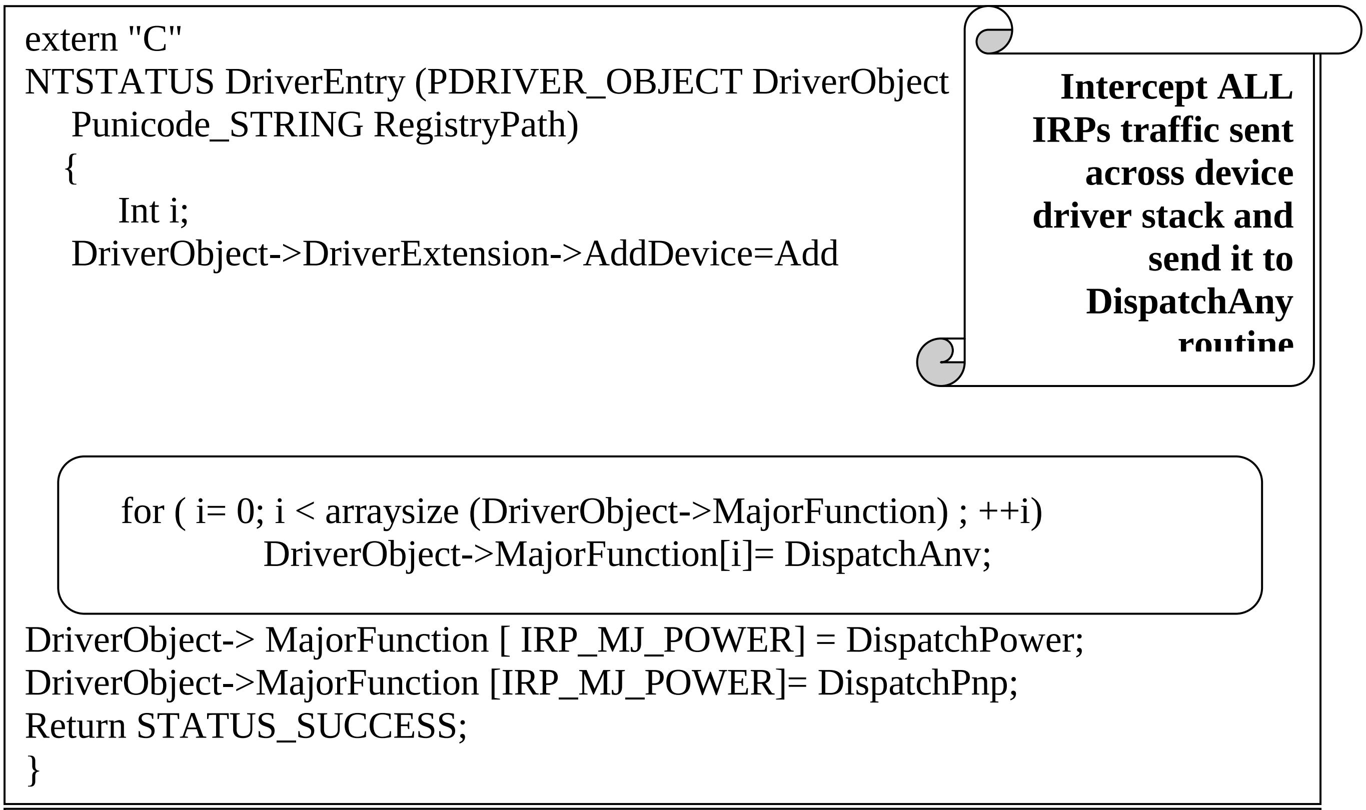


Figure 2 : Shows the communication scheme between the application and the

Figure (2) shows the life cycle of a communication session between user mode windows application and the hardware level device. The IRP constructed by the I/O manager will encapsulated both requested operation (i.e., READ, WRITE) and the data to be sent to the device. [5]

Filter driver will intercept all IRPs exchanged between windows application and the hardware device and it is up to the designer to decide either to manipulate the data within the IRP or just sent down the stack [3,5]

Figure (3) shows how filter driver will direct all IRPs passed through to a special handling function (for example DispatchAny), this handling function will receive all IRPs sent by the application and the operating system to the hardware device (for example USB device).



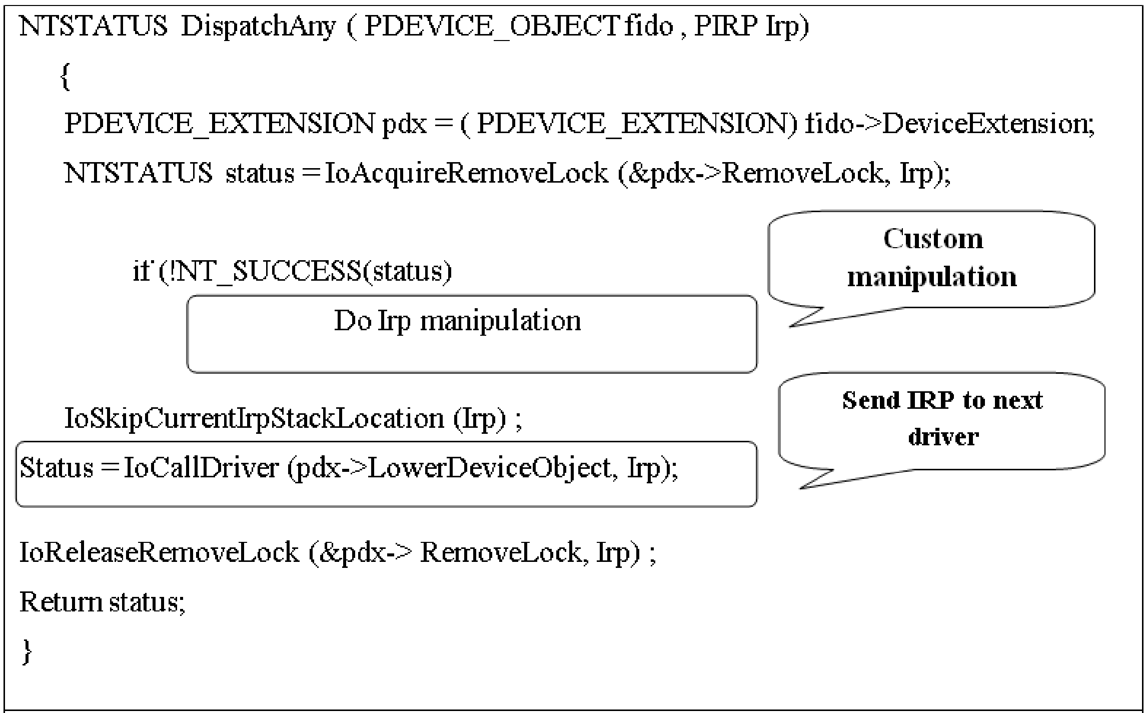
```
extern "C"
NTSTATUS DriverEntry (PDRIVER_OBJECT DriverObject
    Punicode_STRING RegistryPath)
   {
        Int i;
    DriverObject->DriverExtension->AddDevice=Add

        for ( i= 0; i < arraysize (DriverObject->MajorFunction) ; ++i)
                    DriverObject->MajorFunction[i]= DispatchAnv;
DriverObject-> MajorFunction [ IRP_MJ_POWER] = DispatchPower;
DriverObject->MajorFunction [IRP_MJ_POWER]= DispatchPnp;
Return STATUS_SUCCESS;
}
```

Intercept ALL IRPs traffic sent across device driver stack and send it to DispatchAny routine

Figure 3: Shows code segment responsible of intercepting and directing IRP traffic

Figure (4) shows a kernel level code segment that represents the filter driver handler function where all IRPs end to. It is the filter driver handler function to decide what to do with the received IRPs, it could be any processing (encryption/decryption, compression, or any other added functionality). Writing filter driver in the first place to modify the behavior of a device in some way. Therefore, it should have a dispatch functions that do something with some of the IRPs that are directed this way. Normally not all IRP types are to be processed so, many are to be passed down the stack.[1,3,5]



```
NTSTATUS DispatchAny ( PDEVICE_OBJECT fido , PIRP Irp)
   {
   PDEVICE_EXTENSION pdx = ( PDEVICE_EXTENSION) fido->DeviceExtension;
   NTSTATUS status = IoAcquireRemoveLock (&pdx->RemoveLock, Irp);

        if (!NT_SUCCESS(status)
            Do Irp manipulation

   IoSkipCurrentIrpStackLocation (Irp) ;
Status = IoCallDriver (pdx->LowerDeviceObject, Irp);

IoReleaseRemoveLock (&pdx-> RemoveLock, Irp) ;
Return status;
}
```

Figure 4: Shows filter code segment represents the handler general structure

Figure (4) shows DispatchAny() function created to receive all IRPs moving down the stack, it is up to the programmer to pass it down or to reject it as a filteration process.

3. The Proposed USB complete protection model

The core idea in this paper is to secure both the USB device and the data stored in that device due to manipulation of power and PNP packets, where the entire communication sessions established between USB plug and play devices and the computer system are monitored and secured by specially designed filter driver.

The security model presented by this paper is not to secure sensitive data stored in a USB device as an isolated offline procedure, in other words data would not be manipulated at the file system level due to the complications of the file system auditing capabilities which are crucially required at this level.

Secure device driving is the ultimate objective of the model presented in this work, where USB devices' traffic is to be manipulated at the packet level before even assembling data file. Windows I/O manager communicates to the mounted hardware devices through its corresponding installed drivers (i.e., provided by the hardware manufacturer). It is the responsibility of the drivers to do the real communication with the hardware while I/O manager responsibility is only to relay IRPs to/from user level applications and device driver stack (i.e., device stack layered device drivers in a sequential manner).

The work introduced by this paper is composed of many software modules to grant developers the ability to develop each component separately, the main components are:

1- USB lower filter driver : to capture USB traffic
2- Kernel level Encryption Unit: to encrypt/decrypt USB traffic

Configuration unit: to configure the encryption algorithm and interact with kernel module.

.

Figure (5) shows the main components of USB complete protection system.

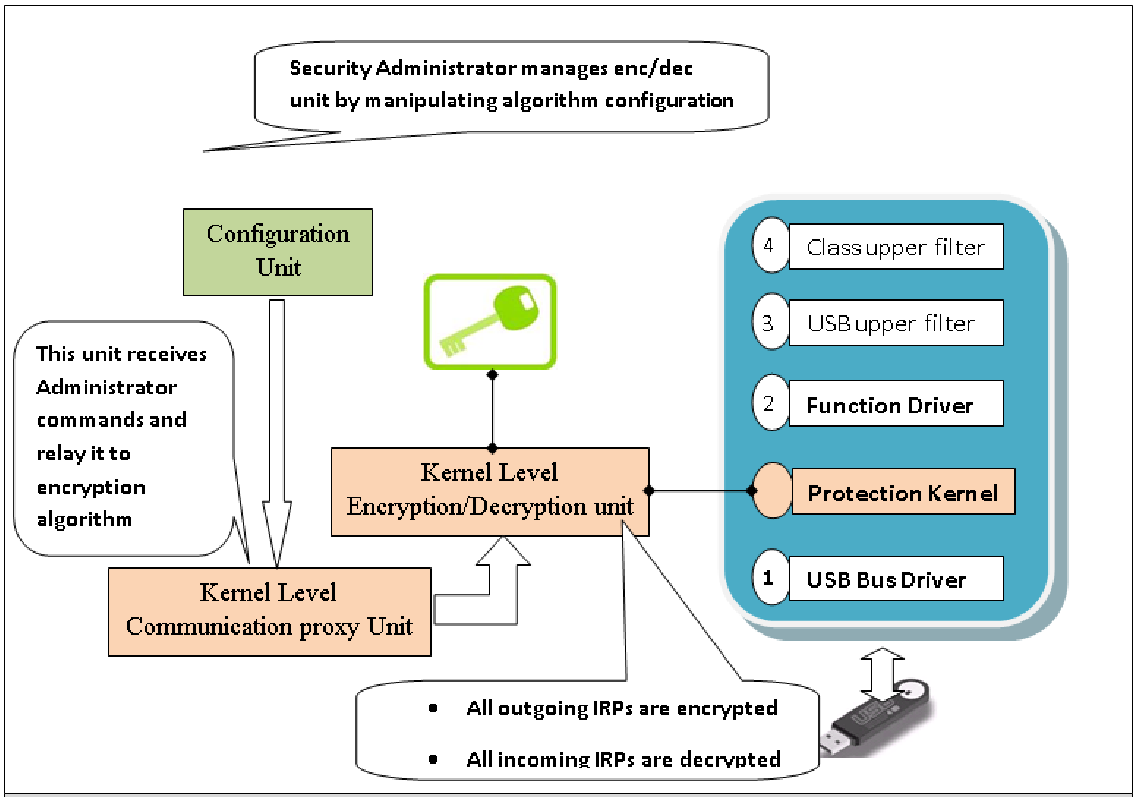

Security Administrator manages enc/dec unit by manipulating algorithm configuration
Configuration Unit
This unit receives Administrator commands and relay it to encryption algorithm
Kernel Level Encryption/Decryption unit
Kernel Level Communication proxy Unit
4 Class upper filter
3 USB upper filter
2 Function Driver
Protection Kernel
1 USB Bus Driver
All outgoing IRPs are encrypted
All incoming IRPs are decrypted

Figure 5: USB complete protection system components

USB protection model presented in this paper is using symmetrical stream cipher algorithm using system of three linear shift registers (LFSR) as figure (6) shows.

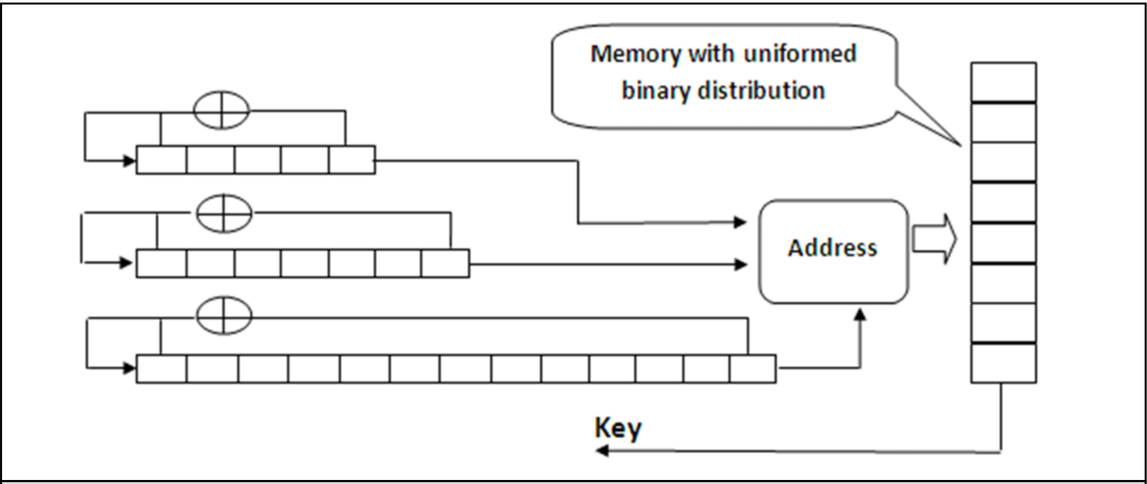
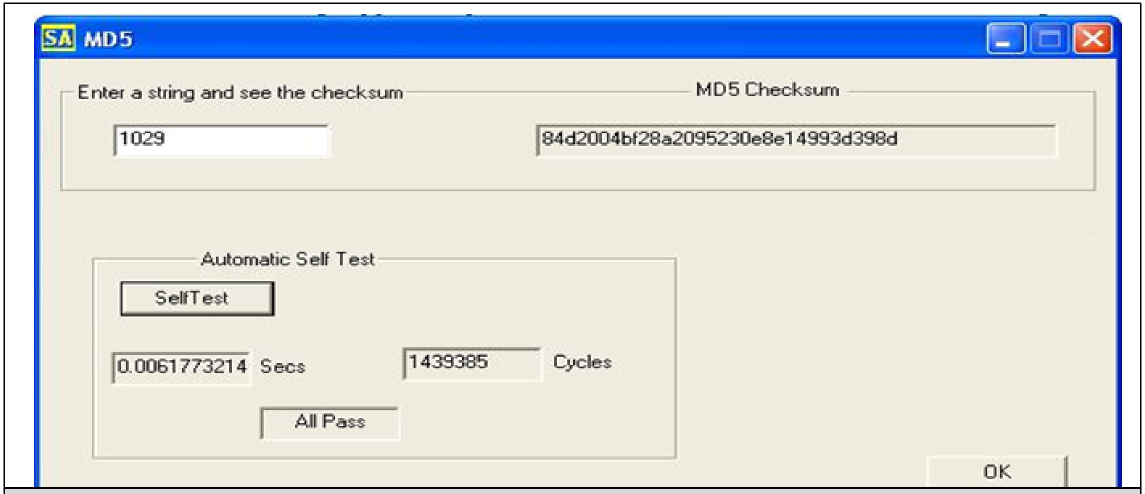


Figure 6: shows the key generator used by USB protection system

The proposed package is upgradable in term of the ability to change the encryption algorithm due to the isolation of operation module and the encryption module.
At each encryption session the round of the key-generator will be registered and saved with the encrypted data in the USB device. This round number could be very useful for the attacker in his/her way to compromise the encryption algorithm, so, this paper also has deployed a security software to encode the round number before saving it with the encrypted data. It is the MD5 module that has been deployed as the figure (7) shows.



1029

84d2004bf28a2095230e8e14993d398d

0.0061773214 Secs

1439385 Cycles

All Pass

Figure 7: shows MD5 in action to encode the round number before saving it with the encrypted

4. USB File Filter Driver

A file system filter driver is a kernel driver that has the same architecture and components of all kernel drivers, the only difference is the processing components where filter driver adds value to or modifies the behavior of a file system which is represented by stream of packets moving up and down the device stack. A file system filter driver is a kernel-mode component that runs as part of the Microsoft Windows NT executive.

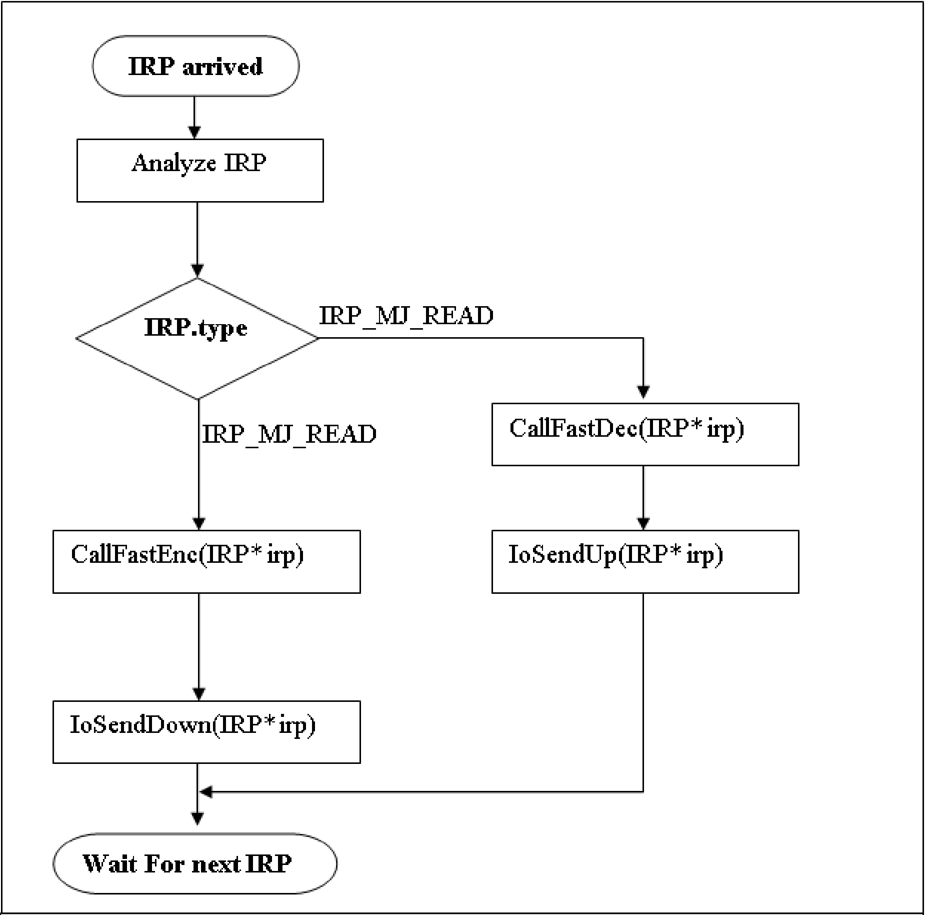


Figure 8: shows flowchart of the USB Protection Lower filter driver

indows operating systems treat all devices as a file to which write/read operations can be conducted. So, the file system filter driver can filter I/O operations for one or more

file systems or file system devices. Depending on the nature of the driver, filter can mean
log, observe, modify, encrypt/decrypt, compress/decompress, or even prevent. Typical applications for file system filter drivers include antivirus utilities, encryption programs, and hierarchical storage management systems.

In this paper filter driver will be used to add security to the file system by encrypting file system send by applications which is encapsulated as stream of IRPs sent by I/O manager.

5. Discussion

All hardware devices visibility is done through the device driver, without a driver it is impossible to communicate the device. In windows many kernels are contacting the device for example I/O manager, PNP manager and Power manager each of which has its own packet and interpretation, the common thing among those kernels is their packets have to pass through device driver, so, any manipulation to entire traffic of IRPs would not affect only the file system or power manipulation but it will affect all system behavior.

The generated information of flash disk will have a great immunity against wide range of viruses due to the fact that viruses have to interpret data before attacking it, otherwise viruses will not be activated, this is the style used be Trojan viruses and other kind.

As it concerns the spyware it has embed itself into the device stack at the authorized machine, and this issue has been resolved by many tapping proof security modules in windows 7. For the flash disk loaded with data generated by this system, the spyware has nothing to get on any other machines because the flash disk can't be read, so, how can the spyware intervene data exchanged with the un authorized system.

6. Conclusions

1- USB devices can be manipulated easily by filter driver without being noticed by traditional anti-virus, this paper has installed USB filter drive and has manipulated the traffic exchanged with the USB device without even a notice from the anti-virus installed on the test machine.

2- USB security system can be guaranteed for the USB device when an attacker tries to get the information on a different machine, so this model presented by this paper can secure the USB device mobility but it can't protect it from other intrusion techniques on the same machine like installing upper level filter driver where information can be captured easily.

7. References

1- Krs: http://www.krs.ca/2011/01/17/malware-virus-now-targets-usb-devices/
2- Sunny: Sunnyvale (California, US) based security firm, threat report, Narus Inc. 2011
3- Microsoft Division, “Microsoft Windows 2000 Driver Design Guide”, Microsoft press,2000
4- Kasper001: www.kaspersky.com/de/news?id=207566365
5- Art Baker and Jerry Lozano, “The Windows 2000 Device Driver Book”, second edition , prentice-Hall, 2001
6- Andrew S. Tanenbaum, “Modern Operating Systems”, second edition , prentice-Hall 2001.
7- Charis Cant, “writing Windows WDM device drivers”, Berkeley, 1999
8- Walter Oney, “Programming the Microsoft Windows Driver Moderl”, second edition ,2003