Edith Cowan University
Research Online

Australian Digital Forensics Conference | Security Research Institute Conferences

2007

Tracing USB Device artefacts on Windows XP operating system for forensic purpose

Victor Chileshe Luo
Edith Cowan University

Originally published in the Proceedings of the 5th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, December 3rd 2007.
This Conference Proceeding is posted at Research Online.
http://ro.ecu.edu.au/adf/23

Tracing USB Device artefacts on Windows XP operating system for forensic purpose

Victor Chileshe Luo
School of Computing and Information Science
Edith Cowan University
vluo@student.ecu.edu.au
cvluo@yahoo.com

Abstract

On Windows systems several identifiers are created when a USB device is plugged into a universal serial bus. Some of these artefacts or identifiers are unique to the device and consistent across different Windows platforms as well as other operating systems such as Linux. Another key factor that makes these identifiers forensically important is the fact that they are traceable even after the system has been shut down. Hence they can be used in forensic investigations to identify specific devices that have been connected to the system in question.

Keywords

USB device identifier, forensic, artefacts, registry key, log file, Windows XP, Operating system

INTRODUCTION

Demand for USB devices such as memory sticks has increased enormously in recent years. In some ways this increase has resulted in more powerful, faster and bigger capacity USB devices. Furthermore USB devices have become more popular in workplaces, education institutions etc. Many employees use them to store company information such as e-mails, corporate documents, third party sensitive data, company directories and business calendars, while Students use them to store assignments, lecture notes and other personal files. USB storage devices can also be used in contrary to the organisation policies. Their size and nature of use sometimes make them suitable to carry out malicious activities. The ability to hold gigabytes of data has certainly introduced considerable security risks, particularly in corporate environments. In addition to providing a means to move data to and from a system, USB storage devices may also be used to introduce malicious code into an otherwise protected system (Gorge, 2005).

However, the popularity or capacity of these devices is not this paper's main focus, but the ability to be able to trace the trails of these tiny devices for accountability. In this paper will discuss how USB storage devices can possibly leave identifiers imbedded within them by manufacturers on Windows XP system.

USB ARTIFACTS

All USB devices have manufacturer's information embedded in them. It is this information that Windows XP operating system uses to build a unique profile that is used to uniquely identify these devices. When these tiny storage devices are attached to a USB port on the system running Windows XP, in-built drivers collect information (manufacturer specifications) from the device and then use that information to create a profile of identifiers. These identifiers end up in different locations on the system and tend to be persistent after shut down (Gorge, 2005). This ability to preserve information about devices reduces reinstallations every time the device is attached to the system. It also increases Windows ability to create profiles of smaller devices such as those devices from same manufacturer.

Proof of consistency

On Linux systems these identifiers are more clear, specific and consistent. Addition information such as manufacturer's name and device description is also clearly identified.

As proof of concept, a Verbatim thumb drive was attached to Linux system (Debian) on two different occasions. The first attachment was an attempt to allow the system to collect relevant information about the device. The second attachment was done at least two weeks after the thumb drive was first attached to the system. The idea of attaching the USB thumb drive a second time was to capture USB information in memory using "cat" command as shown in figure 1 and to ensure the information belonged to the currently attached USB thumb drive.

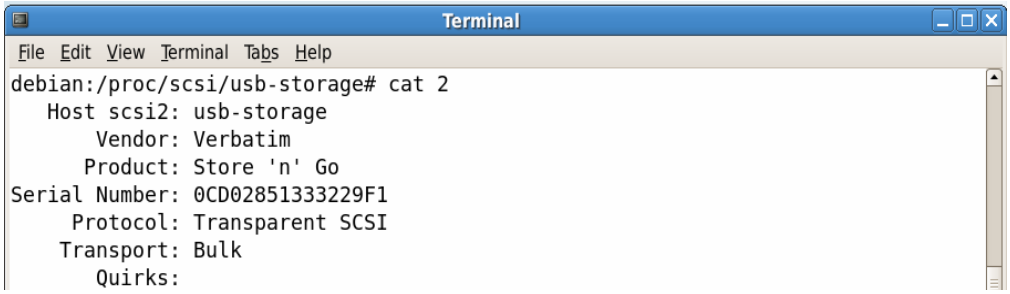

Figure 1. Cached USB identifiers on Linux system

The information collected was then used to locate and compare similar information from log files such as masseges.log and syslog.log. By comparing information in figure 1 and 2, information such as serial number, abbreviation of manufacturer name (VBTM for Verbatim) and product name (Store_n_Go) was successfully found dating back to two weeks. This information was not only well preserved, but also matched the information collected from Windows XP system on the same thumb drive. The outlined discovery is a clear indication that some form of profile is created and preserved every time a new device is attached to the system.

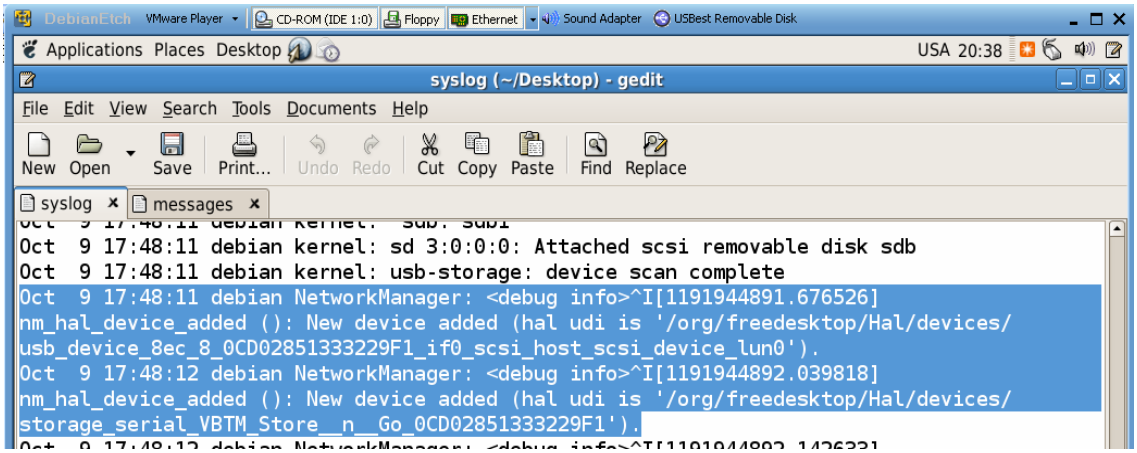

Figure 2. syslog file on Linux system showing the logged USB identifiers

WINDOWS XP APPLICATION

Windows USB identifiers

Windows XP operating system uses USB hub drivers to detect newly installed or attached USB device. When a device is attached to a port, the Windows operating system finds the appropriate driver to read and collects descriptors from it. Then the operating system uses the descriptors to build a unique profile for the device. Information collected is then used by the operating system to find the appropriate driver for the device. To achieve this, the operating system attempts to find device ID in usbstor.inf for those explicitly supported devices. If the USB hub driver enumerates one of these devices, the system will automatically load the USB storage port driver (Microsoft, 2007).

The device IDs for USB mass storage devices listed in usbstor.inf take the usual form for USB device IDs composed using information in the USB device's device descriptor. On Windows XP, a complete device unique identifier takes the following format: USB\VID_*v(4)*&PID_*d(4)*&REV_*r(4)*. According to Microsoft cooperation, *v(4)* is the 4-digit vendor code that the USB committee assigns to the vendor, *d(4)* is the 4-digit product code that the vendor assigns to the device, and *r(4)* is the revision code (Microsoft, 2007). This can be illustrated using the device instance ID from the figure 3: USB\VID_08EC&PID_0008\0CD02851333229f1, where 08EC is the vendor code, 0008 is the product code and 0CD0 is the revision code. All the three descriptors form a unique ID called Device Instance ID.

Figure 3. USB Device Instant ID as shown in device manager

According to Carvey and Altheide Windows also queries the device descriptor for class code (bDeviceClass field), subclass code (bDeviceSubClass field) and protocol code (bDeviceProtocol field) in order to develop a list of compatible Device identifiers (Carvey & Altheide, 2005). The general descriptors Windows uses to generate a profile for a device is shown in figure 4.

Offset	Field	Size	Value	Description
0	bLength	Byte	12h	Size of this descriptor in bytes
1	bDescriptorType	Byte	01h	DEVICE descriptor type
2	bcbUSB	Word	????h	USB specification release number in binary-coded decimal (i.e. 2.10 = 210h). this filed identifies the release of the USB specification with which the device and is descriptors are compliant
4	bDeviceClass	Byte	00h	Class is specified in interface descriptor by USB working
5	bDeviceSubClass	Byte	00h	Subclass is specified in interface descriptor by USB working group
6	bDeviceProtocol	Byte	00h	Protocol is specified in interface descriptor by USB working
7	bMaxPacketSize0	Byte	??h	Maximum packet size for endpoint zero. (only 8, 16, 32, or 64 are valid (08h, 10h, 20h, 40h)
8	idVendor	Word	????h	Vendor identifier (assigned by the USB-IF)
10	idProduct	Word	????h	Product identifier (assigned by the manufacturer)
12	bcdDevice	Word	????h	Device release number in binary-coded decimal
14	iManufacturer	Byte	??h	Index of string descriptor describing the manufacturer
15	iProduct	Byte	??h	Index of string descriptor describing this product
16	iSerialNumber	Byte	??h	Index of string descriptor describing the device's serial number
17	bNumberConfigurations	Byte	??h	Number of possible configurations

Figure 4. A profile of identifiers Windows uses to uniquely identify a device (USB, 1999).

Registry as a USB log file

Anyone looking into Windows registry for forensic purpose must understand that Windows registry is a repository of all information about all aspects of the computer, which includes the hardware, operating system, applications and users. In general, the investigator must be clear of what to look for and where to look for it. In terms of the USB, Windows registry stores information that ensures proper USB devices drivers are loaded, services required by applications are made available, proper application is loaded to open a file when you double click on the icon in the explorer, and that an application window appears in the proper place on your screen when you first launch it (Mee, Tryfonas, & Sutherland, 2006).

USB connections history in the registry is maintained under the following key:

HKEY_LOCAL_MACHINE\System\ControlSet00x\Enum\USBSTOR

The ControlSet in use by the system depends upon the data associated with the following registry value:

HKEY_LOCAL_MACHINE\System\Select\Current (Carvey, 2005).

Every USB device currently and previously connected to system has the device instance identifier listed under USBSTOR key as shown in figure 5.

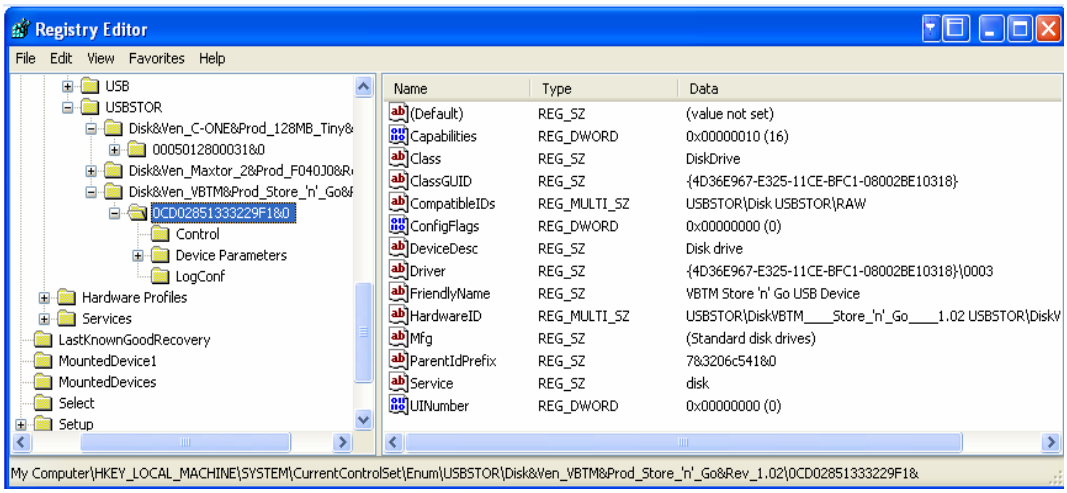

Figure 5. view USB unique ID entry under USBSTOR entry key

The highlighted entry in figure 5 is a unique device identifier, and also a unique serial number for that particular device assigned by the manufacturer. From the findings explained earlier in the paper, this number remains consistent across platforms.

According to Carvey, not all thumb drives will have serial numbers registered in the registry. Some thumb drives are manufactured without serial numbers. If the second character of the unique instance ID is a '&', then the ID was generated by the system (Carvey, 2005).

Another important registry entry is HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion. This key contains specific information about the location of plug and play device .inf files. The information to locate the .inf file is defined in DevicePath value which holds REG_EXPAND_SZ data types. REG_EXPAND_SZ is expandable, capable of holding multiple paths for the DevicePath. (Carvey & Altheide, 2005).

DevicePath registry key list of paths is used by plug and play manager to match the device identifiers with driver ranking the lowest on a scale of 0 to 0xFFFF. Once the driver is identified and loaded, the plug and play (PnP) uses the driver to retrieve any descriptors from the device and attempts to match them with explicitly supported device identifiers in the usbstor.inf. If the match is found, the usbstor.sys driver is installed and creates a new physical device object for each of the device's logical units. The newly formed physical device object has the following format: USBSTOR\v(8)p(16)r(4). To the PnP manager the PDO format is interpreted as v(8) for 8-character vendor identifier, p(16) for 16-character product identifier, and r(4) for 4-character revision level value (Microsoft, 2007).

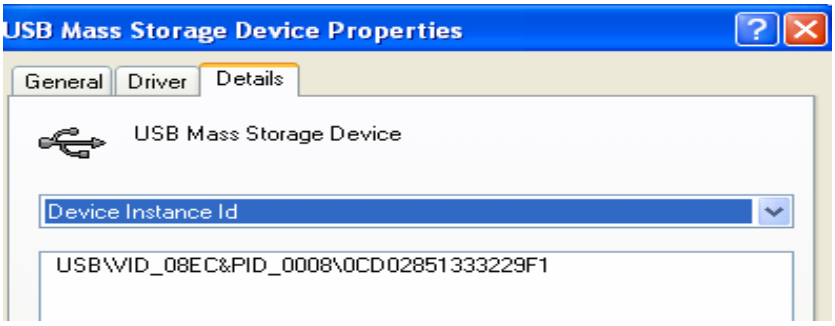

figure 6. View device manufacturer serial number via Device Manager

When PDO of a USB storage device is viewed under device manager, additional 12 characters may be appended to the end of device ID. This is the serial number of the device and the index to this serial number is found in iSerialNumber, which is a value contained in device descriptor. If the value for iSerialNumber is 0x00, then the

device was not assigned serial number by its manufacturer. This 12 character number is unique and persistent across platforms, but the inclusion of this unique identifier in the device is optional as per USB specification (Carvey & Altheide, 2005).

Devices that do not have serial numbers are assigned a 12 character sequence number. This number contains an "&" character and the final value corresponds to the USB port to which the device is connected. The 12 character sequence generated by PnP manager, hence changes when the device is plugged to a different system.

In addition to these device identifiers, usbstor.inf contains compatible class identifiers for each USB based device. These devices can be CD-ROM devices, removable media devices or generic SCIS media devices. During installation these devices can classified under any of the following classes and subclasses:

USB\CLASS_08&SUBCLASS_02&PROT_50

USB\CLASS_08&SUBCLASS_05&PROT_50

USB\CLASS_08&SUBCLASS_06&PROT_50

All devices are firstly classified as mass storage devices (class 08h), then matched with appropriate subclass where subclass 02h is matched with SFF-8020i ATAPI CD-ROM devices, while subclass 05h is matched with SFF-8070i ATAPI removable media and subclass 06h is matched generic SCSI media. Protocol 50h simply means the devices attached are bulky-only transport protocol. According to the results from the investigation carried out earlier, the data retrieved from the USB storage device descriptor must match the USB\CLASS_08&SUBCLASS_06&PROT_50 for the system to load usbstor.sys (Microsoft, 2007).

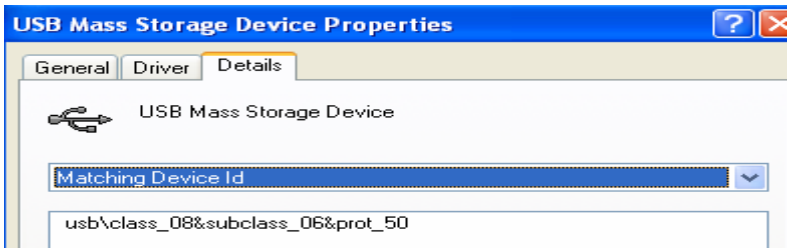

Figure 7. shows a class match for USB storage device

An example of these class and subclass identifiers can be viewed from device manager. While a USB storage device is connected to USB port, open the device manager, under the Universal Serial Bus Controller, right-click on USB Mass Storage Device and choose properties from the drop-down menu, then choose the Details tab, and select "Matching Device ID" from the drop-down menu and the corresponding value will appear below as shown in figure 7.

When compatible USB storage devices are connected to the Windows system, their artefacts are visible in Windows registry and log files. Under HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum\USB registry key, evidence of subkeys representing device IDs of similar format can be easily identified. More subkeys representing instance IDs follow under each subkeys identifying devices that have been connected to the system. Another important registry key for more analysis is:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum\USBStor

USBStor key is similar to the device ID subkeys beneath the USB key, but values under USBStor are in human readable format while values under USB key are in hexadecimal format. As compared to the amount of subkeys under USB key, which is generally for all USB-connected devices, USBStor has fewer subkeys and specifically for USB mass storage devices (Carvey & Altheide, 2005). Beneath this key are several instance ID subkeys, representing each devices that have been connected to the system as shown figure 5.

Associating the timeline of the USB connections with user activities involving USB storage devices is important during registry analysis. When an entry is created in the registry, each keys found under that entry has a value associated with it called "LastWrite" time. This value represents the last time the registry key was modified. During forensic investigation of a USB storage device, the LastWrite times of the keys can be used to determine the timeline with respect to user activities involving USB storage devices (Carvey & Altheide, 2005).

Another interesting entry in the registry is HKEY_LOCAL_MACHINE\SYSTEM|MountDevices\. This particular key provides information about the drive letters association with the devices. The value in ParentidPrefix which is found under MountDevices key can be used to exactly determine or map to the MountedDevices Registry in order to identify the drive letter to which the device was mounted. Beneath the MountedDevices registry key are several values in binary or REG_BINARY data types as shown in figure 8.

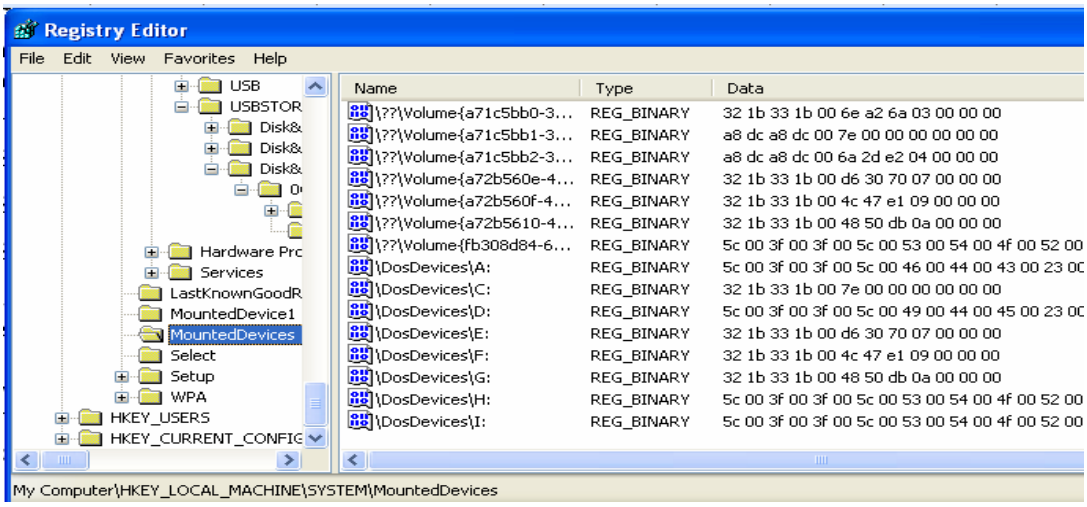

Figure 8 MountedDevices registry keys showing drive letters and unique binary

However, some of the values start with \DosDevices\ followed by drive letter e.g. \DosDevices\H. To find out, Right click on one of them and choose modify. In the "Edit Binary Value" dialog on right-most column, appears characters like this:

\??\STORAGE#RemovableMedia#7&e3d6b7b&0&RM&{53f56307-b6bf-11d0-94f2-00a0c91efb8b}

The 7&e3d6b7b&0&RM portion of the right-most columns is the ParentidPrefix for the device. Using this ParentidPrefix we can determine the last time the device was connected to the system. To do so navigate to the following registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\control\DeviceClasses

Clicking on key identical to 53f56307-b6bf-11d0-94f2-00a0c91efb8b taken from the right-most column of the "Edit Binary Value" dialog box , reveals information about several USB devices that have been attached to the system before as shown in figure 9.

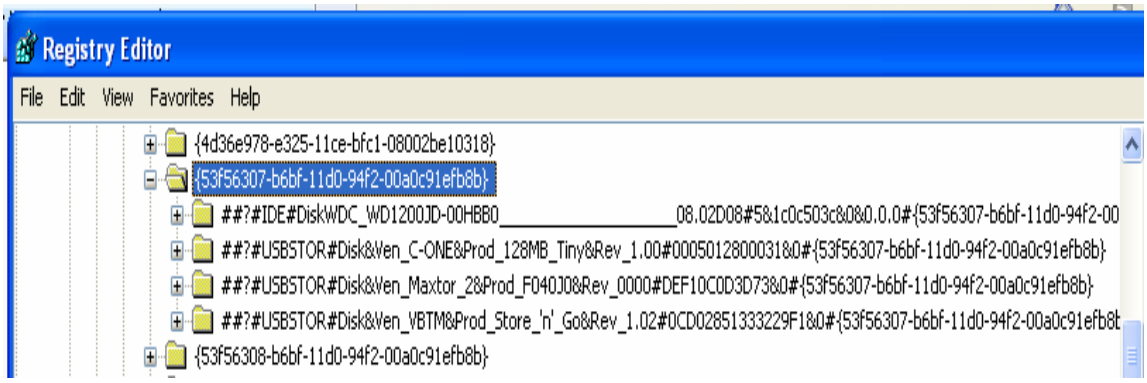

Figure 9 shows devices under DeviceClass registry key

Looking at the last subkey for the highlighted registry key in figure 9, clearly shows the unique instance identifier (OCD02851333229F1) for a USB storage device with product ID "Store_n_Go and manufacturer ID VBTM which is an abbreviation for Verbatim. The portion after unique instance ID (product serial number) is the ParentidPrefix value for the device (Forensic-Wiki, 2006).

To determine the LastWrite time for a specific USB device, open the registry (Click Start, Run and type Regedit.exe), navigate to the USB device key, from the file menu, click "Export", in the "Save As" type drop-down menu, select "Text Files (*.txt), then type the file name and press "Enter". Open the text file using Notepad, and look at the last write time value as shown in figure 10 (Winhelponline, 2007).

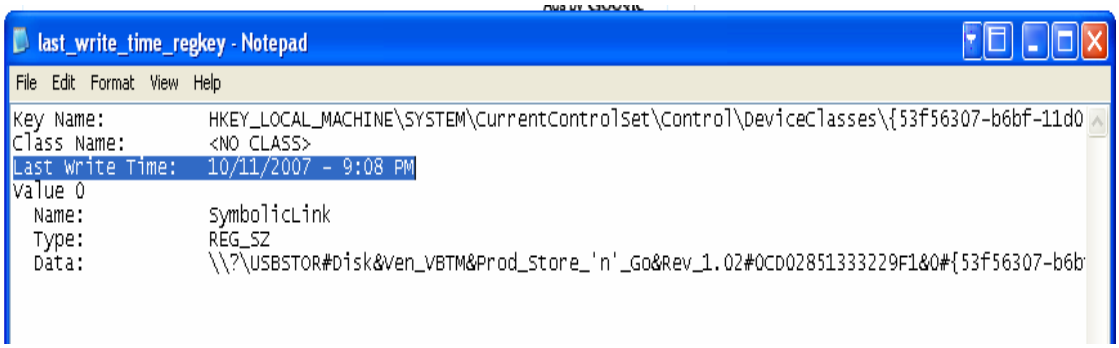

Figure 10 showing the last write time exported to text file from registry

Windows Log Files

Windows log files can help in reinforcing the information collected from the registry. The log file of interest is setupapi.log which is found in %SYSTEMROOT% (C:\WINDOWS on the standard Windows XP install). Every installation of hardware drivers on the system is recorded in this file (Carvey & Altheide, 2005). After installing Store_n_Go USB storage device the setupapi.log recorded the following activities:

> #I306 DICS_START: Device has been started.
>
> [2007/09/30 12:27:03 1496.3 Driver Install]
>
> #-019 Searching for hardware ID(s): usb\vid_08ec&pid_0008&rev_0100,usb\vid_08ec&pid_0008
>
> #-018 Searching for compatible ID(s): usb\class_08&subclass_06&prot_50,usb\class_08&subclass_06,usb\class_08
>
> #-198 Command line processed: C:\WINDOWS\system32\services.exe
>
> #I022 Found "USB\Class_08&SubClass_06&Prot_50" in C:\WINDOWS\inf\usbstor.inf; Device: "USB Mass Storage Device"; Driver: "USB Mass Storage Device"; Provider: "Microsoft"; Mfg: "Compatible USB storage device"; Section name: "USBSTOR_BULK".
>
> #I023 Actual install section: [USBSTOR_BULK.NT]. Rank: 0x00002000. Effective driver date: 07/01/2001.
>
> #-166 Device install function: DIF_SELECTBESTCOMPATDRV.
>
> #I063 Selected driver installs from section [USBSTOR_BULK] in "c:\Windows\inf\usbstor.inf".
>
> #I320 Class GUID of device remains: {36FC9E60-C465-11CF-8056-444553540000}.
>
> #I060 Set selected driver.
>
> #I058 Selected best compatible driver.
>
> #-166 Device install function: DIF_INSTALLDEVICEFILES.
>
> #I124 Doing copy-only install of "USB\VID_08EC&PID_0008\0CD02851333229F1".
>
> #-166 Device install function: DIF_REGISTER_COINSTALLERS.
>
> #I056 Coinstallers registered.
>
> #-166 Device install function: DIF_INSTALLINTERFACES.
>
> #-011 Installing section [USBSTOR_BULK.NT.Interfaces] from "c:\Windows\inf\usbstor.inf".
>
> #I054 Interfaces installed.
>
> #-166 Device install function: DIF_INSTALLDEVICE.
>
> #I123 Doing full install of "USB\VID_08EC&PID_0008\0CD02851333229F1".
>
> #I121 Device install of "USB\VID_08EC&PID_0008\0CD02851333229F1" finished successfully.

On line number I306, the setupapi.log file recorded the time and date the device driver installation began, while on very last line shows that the device was successfully installed. By comparing the installation date from line I306 of the setupapi.log file and the LastWrite time in the registry, it is possible to determine when the device was first connected to the system and for how long the activities might have been repeated. On line I022, the

setupapi.log file recorded more vital information, which is the USB\Class_08&SubClass_06&Prot_50. Subclass 06h in Windows XP system is a predefined driver for generic SCSI media; in this case the USB storage successfully installed and indentified with device instance ID or serial number 0CD02851333229F1 on line I121.

CONCLUSION

The unique identification numbers imbedded in some devices by manufacturer are returned as iserialNumber values on Windows XP system. These unique identifications should be noted to be persistent across identified platforms. The finding raises some interesting issues, for example, an administrator could gather information of good known authorised devices that have been attached to the system. From gathered information, an administrator can determine if any unauthorised USB based storage device has been installed on the restricted machine.

Investigation techniques discussed in this paper cannot only help solve USB storage related cases such information stealing, but can strongly help law enforcers have an idea of how other crimes unrelated to one discussed were committed. In explicitly material investigations, forensic investigators could equip law enforcers with information from setupapi log file showing potential devices used when committing such horrific crimes. The type of drivers installed and identifiers associated with the drivers could help identify specific devices once attached to the system in question. The following setupapi log file shows an artefact depicting a digital camera installation:

[2007/10/11 18:27:16 1488.3 Driver Install]

#-019 Searching for hardware ID(s): usb\vid_040a&pid_05bd&rev_0100,usb\vid_040a&pid_05bd

#-018 Searching for compatible ID(s): usb\class_06&subclass_01&prot_01,usb\class_06&subclass_01,usb\class_06

#-198 Command line processed: C:\WINDOWS\system32\services.exe

#I022 Found "USB\VID_040A&PID_05bd" in C:\WINDOWS\inf\oem18.inf; Device: "KODAK Digital Camera"; Driver: "KODAK Digital Camera"; Provider: "Eastman Kodak"; Mfg: "Kodak"; Section name: "UsbScan.Camera".

#I023 Actual install section: [UsbScan.Camera]. Rank: 0x00000001. Effective driver date: 06/14/2002.

#I393 Modified INF cache "C:\WINDOWS\inf\INFCACHE.1".

#I022 Found "USB\Class_06&SubClass_01&Prot_01" in C:\WINDOWS\inf\ptpusb.inf; Device: "Digital Still Camera"; Driver: "Digital Still Camera"; Provider: "Microsoft"; Mfg: "Generic"; Section name: "PTP".

#I023 Actual install section: [PTP]. Rank: 0x00002000. Effective driver date: 07/01/2001.

#-166 Device install function: DIF_SELECTBESTCOMPATDRV.

#I063 Selected driver installs from section [UsbScan.Camera] in "c:\Windows\inf\oem18.inf".

#I320 Class GUID of device remains: {36FC9E60-C465-11CF-8056-444553540000}.

#I060 Set selected driver.

#I058 Selected best compatible driver.

#-166 Device install function: DIF_INSTALLDEVICEFILES.

#I124 Doing copy-only install of "USB\VID_040A&PID_05BD\C713_0C0390345".

#-166 Device install function: DIF_REGISTER_COINSTALLERS.

#I056 Coinstallers registered.

From the log file, forensic investigators could use line #-019 to determine the type device being installed at that time and the time the installation started by referring to line above it. Line #I022 could help in depicting specific device installed including manufacturer name; in this case KODAK camera was clearly recorded with detailed information attached to it. Forensic investigators could identify specify device by using its unique ID as shown in line #I124.

To law enforcers this evidence could help answer their many questions such as whether the system was used as a storage media for criminal data or perhaps the device at the centre of an investigation might have been used to commit crime.

REFERENCES

Carvey, H. (2005). The Windows Registry as a forensic resource Retrieved 9 October, 2007, from http://0-www.sciencedirect.com.library.ecu.edu.au/science?_ob=ArticleURL&_udi=B7CW4-4GX1J3B-1&_user=1385697&_coverDate=09%2F30%2F2005&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000052520&_version=1&_urlVersion=0&_userid=1385697&md5=f4f6c35575ded24887ccff6cdad1bc5c

Carvey, H., & Altheide, C. (2005). Tracking USB storage: Analysis of Windows artifacts generated by USB storage devices. Retrieved 2 October, 2007, from http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B7CW4-4G82Y3M-1&_user=10&_coverDate=06%2F30%2F2005&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=14db0715620630bcf24ee0ced035f073

Forensic-Wiki. (2006). USB History Viewing. Retrieved 15 October, 2007, from http://www.forensicswiki.org/wiki/USB_History_Viewing

Gorge, M. (2005). USB & other portable storage device usage. Retrieved 9 October, 2007, from http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6VNT-4GY9043-8&_user=10&_coverDate=08%2F31%2F2005&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=57444a1440590bffc1945e26c93eee02

Mee, V., Tryfonas, T., & Sutherland, L. (2006). The Windows Registry as a forensic artefact: Illustrating evidence collection for Internet usage Retrieved 10 October, 2007, from http://0-www.sciencedirect.com.library.ecu.edu.au/science?_ob=ArticleURL&_udi=B7CW4-4M0S394-1&_user=1385697&_coverDate=09%2F30%2F2006&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000052520&_version=1&_urlVersion=0&_userid=1385697&md5=e5322a5cb4f4119534e0a0273159db63

Microsoft. (2007). Identifiers Generated by USBSTOR.SYS. Retrieved 10 october, 2007, from http://msdn2.microsoft.com/en-us/library/ms791086.aspx

USB. (1999). Universal Serial Bus Mass Storage Class Bulk-Only Transport. Retrieved 9 October, 2007, from http://www.usb.org/developers/devclass_docs/usbmassbulk_10.pdf

Winhelponline. (2007). Determining the "Last Write Time" of a registry key? Retrieved 15 October, 2007, from http://www.winhelponline.com/articles/12/1/

COPYRIGHT

Victor Chileshe Luo ©2007. The author/s assign Edith Cowan University a non-exclusive license to use this document for personal use provided that the article is used in full and this copyright statement is reproduced. Such documents may be published on the World Wide Web, CD-ROM, in printed form, and on mirror sites on the World Wide Web. The authors also grant a non-exclusive license to ECU to publish this document in full in the Conference Proceedings. Any other usage is prohibited without the express permission of the authors.