ISSN:0975-9646

Windows Registry Forensics: An Imperative Step in Tracking Data Theft via USB Devices

Tanushree Roy, Aruna Jain

Department of I.T.
Birla Institute of Technology, Mesra, Ranchi, India

***Abstract*— Owing to the increasing pace of occurrence of crimes in digital world, cyber forensic investigation is becoming a burning topic in the field of information security. Registry is an important location in Windows system that contains footprints of user activities and other configuration data, which may be valuable for forensic investigators in collecting potential evidences from the system. This work aims to point out the significance of Registry Analysis, and attempts to answer why it should be carried as a part of digital forensic investigation by demonstrating the role played by Registry in tracking data theft from system to USB external devices.**

***Keywords*— Forensics Analysis, Registry Analysis, Tracking Data Theft, USB footprints, Windows Forensics**.

I. INTRODUCTION

In the recent years, with the development of Information and Communication Technology (ICT) and rise in Internet usage, society's dependence upon computers is increasing rapidly. With this growth of technology, the world is also seeing a substantial rise in the abuse of various kinds conducted with, through or by technology. According to the Norton Cyber Crime report-2011 by Symantec Corporation [9], about *1 million cyber crimes are occurring every day across the globe*. According to the report, the total number of victims in India is 29.9 million, which is approx. 80% of the online adults. The reason for this increasing abuse is attributed to the still-to-be-mature existing security procedures and the reluctance persistent among users in employing security methods as an integral part of the whole system.

As a result, cyber crimes are increasing and, cyber criminals are growing in sophistication as technology acts as a boon for them too. Thus, it becomes exceptionally critical for the law enforcement officers and incident responders to understand computer systems and be able to examine them effectively and efficiently.

Cyber forensics is the branch of science that acts as a tool for the investigators for investigating a computer system or network alleged of being involved in criminal activity and, gathering artifacts that may be used as evidence in the case and presented in the court of law.

Due to its effective GUI and ease of use, Microsoft Windows is one of the most popular operating system and, is; unfortunately the most attacked one too. As windows source code is unavailable, forensic analysis of windows systems becomes a challenging task for the investigators.

Registry is one of the areas in a Windows system where evidences can be found. This work aims to point out the importance of Registry Analysis process carried in Windows Systems as a part of digital forensic investigation in today's scenario. An offline registry parser developed as a part of this work will be used to generate registry keys from registry hives, extracted from the hard disk as a part of postmortem analysis.

In this Information age, ownership of intellectual property is very precious and prized. Theft of intellectual property is one of major issues of concern, which can become a trouble not only for an individual, but for a whole organization. USB ports, as well as other ports that permits one to attach a removable storage device, can act as a promising means to steal classified information. Any user with a removable USB drive can attach the device to the system and copy critical information.

In this paper we have discussed how by means of a careful investigation of the Registry files, data transfer to USB devices be identified. We begin by stating the work done by various researchers in section 2, and explain the structure of the Windows Registry and how Registry tree structure is parsed from the hive file in an offline mode in section 3. In section 4 we will briefly discuss the footprints left on the system and Registry when a USB device is connected. We finally show how to proceed in a case involving data transfer from system to USB through Registry analysis.

II. RELATED WORK

During the past years, it has turn out to be absolutely lucid that Registry in Windows systems contains ample amount of information for the use of incident responders and forensic analysts alike. A great deal of information on how to interpret various Registry data and settings have been provided by Carvey [1], Wong [13] and Farmer [4]. As illustrated by Carvey, "Registry data consists of a wealth of information that the investigator can make use of to make up his case". Kim, et. al. [6] has listed some registry keys that an investigator must check during an investigation. Chang, et. al. [2] has shown how to proceed in an investigation involving Windows systems, and listed some of the registry areas that need attention in the preliminary stages. Additional areas vital from a forensic viewpoint apart from Registry in Windows systems have been noted by Dashora, et. al. [3], such as event logs, RAM, Pagefile, slack space, etc.

Manchanda, et. al. [7] have illustrated how the last-write times associated with every Registry key can be useful in generating a forensic timeline of the events that has occurred in a system.

Documentation regarding the Registry internal structure has been provided in detail by Russinovich [10]. Morgan [8] provided a comprehensive description of the Registry's internal data structures and format that is helpful in generating the registry tree from hive files.

III. METHODOLOGY

A. The Windows Registry

Windows Registry is the "central hierarchical database" used to store information that is necessary to configure the system for one or more users, applications, and hardware devices [11]. It is the *Heart and soul* of Windows operating system; every application that runs on Microsoft's operating systems do absolutely nothing without consulting the Registry first. When we double-click over a file, Windows consults the Registry to figure out what to do with that file. When a new device is connected to the system, Windows assigns resources to the device based on information in the Registry and then stores the device's configuration in the Registry.

1) *Registry Structure Overview*: The Registry is organized in a tree like structure which is equivalent to the filesystem. For e.g., the keys and subkeys found within the five main hives are comparable to folders and subfolders of Windows filesystem, and a key's value is similar to a file inside a folder, a value's *name* is analogous to a filename, its *type* resembles a file extension, and its *data* is like to the actual contents of a file. The registry structure is illustrated in figure 3.1.

The registry value contains 3 parts; value name, value type and value data; as illustrated in figure 3.2.

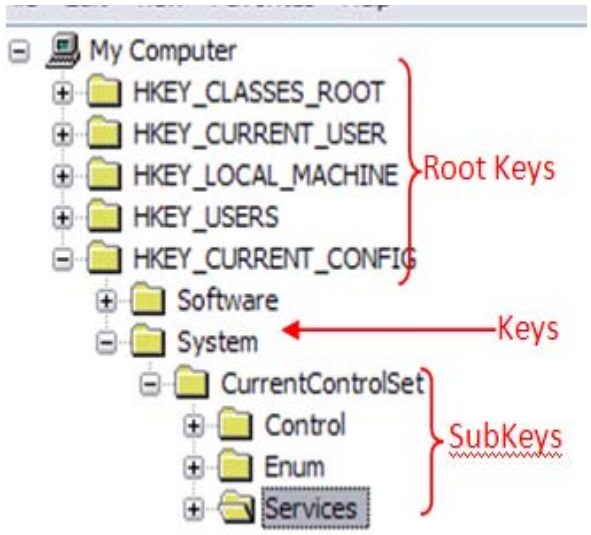

Figure 3.1: Registry Logical Structure as displayed by the Registry Editor regedit.

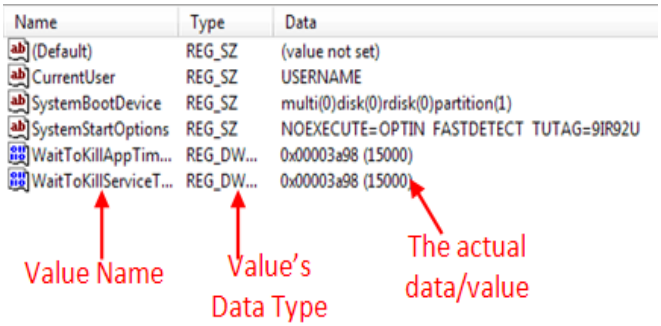

Figure 3.2: Value Fields

Physically, the Registry isn't simply one large file but rather a set of discrete files called *hives*. Each hive contains a Registry tree that has a key serving as the root or initial point of the tree. Subkeys and their values are below the root. Table 1 below lists the Registry hives and their on-disk location for a Windows XP system. The path names of all hives excluding for user profiles are coded into the configuration manager.

Registry Hive Path	Hive file path
HKLM\SAM	%SystemRoot%\System32\Config\sam
HKLM\ SECURITY	%SystemRoot%\System32\Config\security
HKLM\ SOFTWARE	%SystemRoot%\System32\Config\software
HKLM\ SYSTEM	%SystemRoot%\System32\Config\system
HKLM\ HARDWARE	volatile hive
HKU\.Default	%SystemRoot%\System32\Config\Default
HKU\<SID of local service account>	\Documents andSettings\%UserProfile% \LocalService\Ntuser.dat
HKU\<SID of network service account>	\Documents andSettings\%UserProfile% \NetworkService\Ntuser.dat
HKU\<SID of username>	\Documents andSettings\%UserProfile% \Ntuser.dat
HKU\<SID of username>_Classes	\DocumentsandSettings\%UserProfile% \LocalSettings\ApplicationData\Microsoft\Windows\UsrClass.dat

Table 1: Registry hive file locations

2) *Registry hive structure*: A hive is logically divided into allocation units called blocks as disk is divided into clusters. Typically a block is of size 4096 bytes (4 KB). The initial block of a hive is the base block. Remaining blocks contains bins, which contain cells.

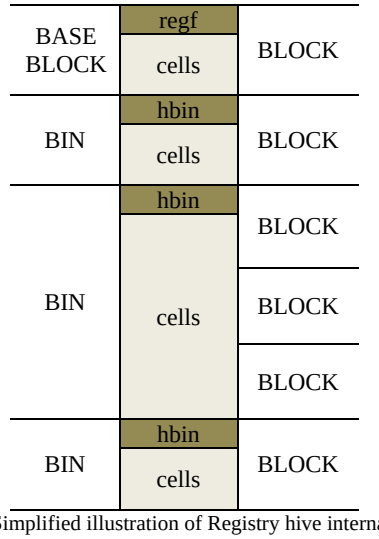

Figure 3.3: Simplified illustration of Registry hive internal structure.

The base block contains global information about the hive, together with a signature—regf, that classifies the file as a hive, sequence numbers, time-stamp that indicates the last time a write operation commenced on the hive, the hive format version number, checksum value, and the hive file's internal file name (for eg., \Device\Harddisk-Volume1\WINDOWS\SYSTEM32\CONFIG\SAM).

Offset	Size	Contents
0x0000	4	"regf" signature
0x000c	8	timestamp
0x0024	4	root key offset
0x0028	4	offset to last hbin in the hive
0x0030	64	file name (Unicode)

*(size in bytes)

Table 2: Important fields of base block

Windows organizes the Registry data that a hive stores in containers called cells. A cell can contain a key, a list of subkeys, a value, a security descriptor, or a list of key values. A field at the beginning of a cell's data describes the data's type. Bins also have headers that have a signature, hbin, a field that contains offset to the hive file of the bin and the bin's size.

Cell Types	Signature	
Key cell	nk	0x6B6E
Value cell	vk	0x6B76
Security Descriptor cell	sk	0x6B73
Sub-key list cell	lf / lh / ri / li	0x666C / 0x686C / 0x6792 / 0x696C
Value-list cell	–	N/A (No header)
Data cell	–	N/A (No header)

Table 3: Cell types and their signatures

3) Traversing the Registry Hive: The base block points to the first hbin structure that in turn comprise the offset to the root key of the registry hive. All key/subkey in the registry hive is having a signature NK. Key cell include an offset to the value-list member containing the values present under the keys and an offset to the related sub-key structure that point to an LH, an LF or an RI cell.

The LH or LF entry includes a member that holds a list of offsets to all the other sub-keys of the key. The value-list contains pointers to value keys. The value entry contains information related to the value present under a key and the associated data. If the data size contains a 1 in its MSB (0x80), the key contains inline data, i.e. – the offset to the data is the actual data rather than an offset to the data. In case of non-inline data, the offset to the data is the offset of the data cell. In case the first two bytes of the data cell contain the signature 'db' (0x6264), it is a data cell with non-contiguous data. This may be because the size of the data is very large, and cannot be accommodated in a single bin. The data is then spread over a number of bins. The next two bytes then tell us how many bins the data is spread over. Following these two bytes is a list of offsets to the data cells that make up this data entry. The data from all these cells can be concatenated to form the actual data of that particular value.

B. Tools Used

1) Collection of Registry hive files: Registry files are opened by the Kernel in restricted mode which means that they cannot be copied while the system is in use, except for the User Registry Files (NTUSER.DAT) that are not currently loaded.

A method was needed for extracting example Registry Files for research use. The chosen method was:

i. Hive files are collected from several machines by booting them from a Linux disc (as hive files are locked while the Windows operating system is running).

ii. ERUNT tool also may be used to obtain hive files from a live system, without shutting-down the system.

It is noted that, a collection method like listed above can capture only the stable hives present in the hard-disk, and not the volatile hives (the in-memory version of disk hives).

2) Registry Examination: Analysis of the registry hive files can be done manually using WinHex [12] or by parsing the hive using an offline parser. For the purpose of examination an offline registry parser `r_parser.pl` was developed in Perl based upon the available knowledge about the Registry structure. The code was further modified to generate specific keys related to USB devices, as explained in the following section. Another utility regkey_time.pl was developed, that listed the keys in a descending order of their last-write times.

C. Footprints left by a USB Device

1) setupapi.log: Since almost all devices now-a-days, are of the type "plug-and-play", containing their associated driver files written on the device firmware, the system can install them directly, ruling out need for a separate installation disk. Whenever such Plug-and-Play USB device is connected to a system, Plug-and-Play (PnP) manager receives this event and queries the device description in its firmware, such as manufacturer, serial no, etc. Upon receiving the information, the PnP manager locates device drivers and a set of Registry keys are created, as described below. Above events are recorded in *setupapi.log* file present in %Windowsdir% (C:\Windows\setupappi.log) when the device gets connected to the system for the first time.

The device detail is not a part of memory area and thus, is not available when we make its clone.

2) Registry Keys created: After the device is identified, a set of registry keys gets created as follows:

i. `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet00x\Enum\USBSTOR\<device_class>\<device_unique_id>\`
ii. `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet00x\Control\DeviceClasses\{<disk_devices_GUID>}\`
iii. `<device_class#device_unique_id#{disk_devicesGUID}>\`
iv. `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet00x\Control\DeviceClasses\{<volume_devices_GUID>}\`
v. `<STORAGE_RemovableMedia#ParentId_Prefix#{volume_devices_GUID}>\`
vi. `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet00x\Enum\Storage\RemovableMedia\<ParentID_Prefix>\`

These keys contain details about the device id, driver description, manufacturer, friendly name, parented prefix, etc. When we connect the same USB to the system again, a sub-key named control is created under the above keys. As a result the time-stamp of these keys reflect the last time the USB was connected to the system.

3) Drive letter to which the device gets mounted: USB device when connected to the system gets assigned to a drive letter (G, H, etc.), which can be identified through the following key: `HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices`

This key contains a value starting with `\??\Volume\` that contains binary data having ParentId_Prefix in the form `<STORAGE_RemovableMedia#ParentId_Prefix#{volume_devices_GUID}>`. This data is also present in the value `\DosDevices\<drive_letter>`, if this USB device was the last device mapped to that drive letter.

4) Finding the user profile through which USB device was connected: The value present in the key `\MountedDevices` starting with `\??\Volume\{…}` occurs only once more in the ntuser.dat hive of the user profile in which the USB device was connected. Using this value, we can find out the user profile through which the USB device was connected.

5) Time the drive was last connected to the system: The first time USB device was connected to a system is found from the *setupapi.log* file and the corresponding registry entries. To associate this time with the actual time, the Registry key `SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones`, present in the Software hive is checked.

6) To track if file opened or copied through explorer: If any file is opened through the explorer by double click on the file name, an entry is created in the Registry key: `\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\`

If file opened using open file menu or saved using save-as file menu in any application program, it is noted in the registry key `\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU` `\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU`

These entries are present in the ntuser.dat registry hive of the user profile identified in 4) above.

7) File copy to USB through other modes: Analysis of the file MAC times: If file is transferred to USB using the copy, cut or send-to context menu option, no registry entry gets created; and one has to examine both the system and USB device file system to track the file copy.

Whenever a file is accessed (i.e., copy, move, open or edit), their MAC (modified, accessed, created) times gets updated. However if the Registry value `NtfsDisableLastAccessUpdate` present in the System hive under the key `ControlSet00x\Control\FileSystem\` is enabled, then MAC times are not updated. By default this value is not present in Windows XP systems. The MAC times of the files suspected of being copied is checked.

8) Analysis of the USB device: Once we have identified the user profile through which the USB device was connected, and it is established that files have been copied to the USB, case can be established against that person, and his USB device is confiscated for analysis. If the copied files are present in the USB device then, their md5 hash values are compared to the values of original files. And if files are not present, then unallocated space is analyzed and files are recovered, if not overwritten till now.

For tracking data transfer from system to USB devices evaluating the performance we connected some USB devices to a system for the first time and copied a few files from system to USB using different modes. To be able to the compare the changes, we took a snapshot of MAC times, before and after files were copied from the system. Table 4 below shows the operations performed on the files and the files' original MAC times. Thereafter, the Registry files of the system were extracted and analyzed using `r_parse.pl`. The analysis process is discussed step by step in section 4 for an USB device.

File name	Date Modified	Date Created	Date Accessed	Operation Performed
Feather Texture.bmp	2/28/2006 2:00 AM	2/28/2006 2:00 AM	12/13/2011 9:21 PM	Copy > Paste
Gone Fishing.bmp	2/28/2006 2:00 AM	2/28/2006 2:00 AM	12/13/2011 9:21 PM	Send-to Option
Green Stone.bmp	2/28/2006 2:00 AM	2/28/2006 2:00 AM	12/13/2011 9:21 PM	Move (Cut >Paste)
ie7beta2.log	11/21/2011 4:47 PM	11/21/2011 4:43 PM	11/21/2011 4:47 PM	Rename
ii6.log	11/21/2011 5:54 PM	4/25/2006 6:00 PM	11/21/2011 5:54 PM	Edit
ie7.log	11/21/2011 5:54 PM	11/21/2011 5:52 PM	11/21/2011 5:54 PM	Open (double click)
Prairie Wind.bmp	2/28/2006 2:00 AM	2/28/2006 2:00 AM	12/13/2011 9:21 PM	Save through Save-As
River Sumida.bmp	2/28/2006 2:00 AM	2/28/2006 2:00 AM	12/13/2011 9:21 PM	Open through explorer

Table 4: Files' original MAC times and operations performed on them

For the sake of simplicity, this work we have assumed that the system fingerprints were not removed deliberately by the user, and are present within the registry and other locations.

IV. Results And Discussions

The result of the systematic analysis for one USB device is follows:

A. setupapi.log file

The setupapi.log file is analyzed to find the entries related to driver installations for the USB device when connected to the system for the first time. The figure 4.1 below shows the entries created for the USB device in question. The device unique instance-id, ParentId prefix and the time-stamp values are highlighted.

```
[2012/05/04 11:51:34 1088.7 Driver Install]
#-019 Searching for hardware ID(s):
usbstor\disksandisk_cruzer_blade____1.00,usbstor\disksandisk_cruz
er_blade____,usbstor\disksandisk_,usbstor\sandisk_cruzer_blade___
_1,sandisk_cruzer_blade____1,usbstor\gendisk,gendisk
#-018 Searching for compatible ID(s): usbstor\disk,usbstor\raw
#-198 Command line processed:
C:\WINDOWS\system32\services.exe
#I022 Found "GenDisk" in C:\WINDOWS\inf\disk.inf; Device: "Disk
drive"; Driver: "Disk drive"; Provider: "Microsoft"; Mfg:
"(Standard disk drives)"; Section name: "disk_install".
#I023 Actual install section: [disk_install.NT]. Rank: 0x00000006.
Effective driver date: 07/01/2001.
#-166 Device install function: DIF_SELECTBESTCOMPATDRV.
#I063 Selected driver installs from section [disk_install] in
"c:\windows\inf\disk.inf".
#I320 Class GUID of device remains:
{4D36E967-E325-11CE-BFC1-08002BE10318}.
#I060 Set selected driver.
#I058 Selected best compatible driver.
#-166 Device install function: DIF_INSTALLDEVICEFILES.
#I124 Doing copy-only install of
"USBSTOR\DISK&VEN_SANDISK&PROD_CRUZER_BLADE&R
EV_1.00\200402033009D8D25377&0".
```

Figure 4.1 (a): Entries of *setupapi.log* for a "SanDisk Cruzer Blade" USB drive with 4GB capacity was connected to the system for the first time.

```
[2012/05/04 11:51:38 1088.11 Driver Install]
#-019 Searching for hardware ID(s): storage\volume
#-018 Searching for compatible ID(s): storage\volume
#-198 Command line processed:
C:\WINDOWS\system32\services.exe
#I022 Found "STORAGE\Volume" in C:\WINDOWS\inf\volume.inf;
Device: "Generic volume"; Driver: "Generic volume"; Provider:
"Microsoft"; Mfg: "Microsoft"; Section name: "volume_install".
#I023 Actual install section: [volume_install]. Rank: 0x00000000.
Effective driver date: 07/01/2001.
#-166 Device install function: DIF_SELECTBESTCOMPATDRV.
#I063 Selected driver installs from section [volume_install] in
"c:\windows\inf\volume.inf".
#I320 Class GUID of device remains:
{71A27CDD-812A-11D0-BEC7-08002BE2092F}.
#I060 Set selected driver.
#I058 Selected best compatible driver.
#-166 Device install function: DIF_INSTALLDEVICEFILES.
#I124 Doing copy-only install of
"STORAGE\REMOVABLEMEDIA\7&251E0F1A&0&RM".
#-166 Device install function: DIF_REGISTER_COINSTALLERS.
#I056 Coinstallers registered.
#-166 Device install function: DIF_INSTALLINTERFACES.
#-011 Installing section [volume_install.Interfaces] from
```

Figure 4.1 (b): Entries of *setupapi.log* for a "SanDisk Cruzer Blade" USB drive with 4GB capacity was connected to the system for the first time

B. Registry entries in SYSTEM and ntuser.dat hive

The registry keys retrieved for the device from the System hive using our parser is shown in figure 4.2.

```
ControlSet001\Enum\USBSTOR\ [Wed Apr 11 17:12:04 2012]
\Disk&Ven_SanDisk&Prod_Cruzer_Blade&Rev_1.00 [Fri May  4 11:51:34 2012]
** \2004020330009D8D25377&0 [Fri May  4 11:51:35 2012]
 [V]-- DeviceDesc (REG_SZ) = Disk drive
 [V]-- Capabilities (REG_DWORD) = 0x00000010 (16)
 [V]-- UINumber (REG_DWORD) = 0x00000000 (0)
 [V]-- HardwareID (REG_MULTI_SZ) = [0] USBSTOR\DiskSanDisk_Cruzer_Blade____1.00
                                   [1] USBSTOR\DiskSanDisk_Cruzer_Blade____
                                   [2] USBSTOR\DiskSanDisk_
                                   [3] USBSTOR\SanDisk_Cruzer_Blade____1
                                   [4] SanDisk_Cruzer_Blade____1
                                   [5] USBSTOR\GenDisk [6] GenDisk
 [V]-- CompatibleIDs (REG_MULTI_SZ) = [0] USBSTOR\Disk [1] USBSTOR\RAW
 [V]-- ClassGUID (REG_SZ) = {4D36E967-E325-11CE-BFC1-08002BE10318}
 [V]-- Service (REG_SZ) = disk
 [V]-- ConfigFlags (REG_DWORD) = 0x00000000 (0)
 [V]-- ParentIdPrefix (REG_SZ) = 7&251e0f1a&0
 [V]-- Driver (REG_SZ) = {4D36E967-E325-11CE-BFC1-08002BE10318}\0110
 [V]-- Class (REG_SZ) = DiskDrive
 [V]-- Mfg (REG_SZ) = (Standard disk drives)
 [V]-- FriendlyName (REG_SZ) = SanDisk Cruzer Blade USB Device
----\Device Parameters [Fri May  4 11:51:38 2012]
----\MediaChangeNotification [Fri May  4 11:51:34 2012]
----\LogConf [Fri May  4 11:51:34 2012]
-------------------------------------------------------------------------------

ControlSet001\Enum\STORAGE\RemovableMedia\ [Fri May  4 11:51:34 2012]
\7&251e0f1a&0&RM [Fri May  4 11:51:35 2012]
 [V]-- Capabilities (REG_DWORD) = 0x00000060 (96)
 [V]-- ConfigFlags (REG_DWORD) = 0x00000000 (0)
 [V]-- HardwareID (REG_MULTI_SZ) = [0] STORAGE\Volume
 [V]-- CompatibleIDs (REG_MULTI_SZ) = [0] STORAGE\Volume
 [V]-- ClassGUID (REG_SZ) = {71A27CDD-812A-11D0-BEC7-08002BE2092F}
 [V]-- Class (REG_SZ) = Volume
 [V]-- Driver (REG_SZ) = {71A27CDD-812A-11D0-BEC7-08002BE2092F}\0109
 [V]-- Mfg (REG_SZ) = Microsoft
 [V]-- DeviceDesc (REG_SZ) = Generic volume
---- \LogConf [Fri May  4 11:51:34 2012]
```

Figure 4.2 (a): Registry values for the USB device in question.

```
ControlSet001\Control\DeviceClasses\{53f56307-b6bf-11d0-94f2-00a0c91efb8b}\ [Fri May  4
11:51:34 2012]
\##?#USBSTOR#Disk&Ven_SanDisk&Prod_Cruzer_Blade&Rev_1.00#200402033009D8D253
77&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b} [Fri May  4 19:22:15 2012]
---- \# [Fri May  4 11:51:35 2012]
 [V]-- SymbolicLink (REG_SZ) =
\\?\USBSTOR#Disk&Ven_SanDisk&Prod_Cruzer_Blade&Rev_1.00#200402033009D8D25377
&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
-------------------------------------------------------------------------------

ControlSet001\Control\DeviceClasses\{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}\ [Fri May  4
11:51:34 2012]
\##?#STORAGE#RemovableMedia#7&251e0f1a&0&RM#{53f5630d-b6bf-11d0-94f2-00a0c91ef
b8b} [Fri May  4 19:22:15 2012]
---- \# [Fri May  4 11:51:35 2012]
 [V]-- SymbolicLink (REG_SZ) =
\\?\STORAGE#RemovableMedia#7&251e0f1a&0&RM#{53f5630d-b6bf-11d0-94f2-00a0c91efb8
b}
```

Figure 4.2 (b): Registry values for the USB device in question.

C. Finding the drive the USB was mounted

The above USB was mapped to E: drive and we got the values as below.

```
MountedDevices\ [Fri May  4 18:51:34 2012]
\DosDevices\E: (REG_BINARY) =
\??\STORAGE#RemovableMedia#7&251e0f1a&0&RM#{53f5630d-b6bf-11d0-94f2-00a0c91efb
8b}
\??\Volume{20275d31-9619-11e1-abc8-0016e693df12} (REG_BINARY) =
\??\STORAGE#RemovableMedia#7&251e0f1a&0&RM#{53f5630d-b6bf-11d0-94f2-00a0c91efb
8b}
```

D. Finding the user profile through which USB device was connected

`\??\Volume{20275d31-9619-11e1-abc8-0016e693df12}` was found in the ntuser.dat hive of the user 1. This means the system user 1 must have connected the USB to the system.

E. Time the drive was last connected to the system

The time USB device was first connected to a system can be found from the *setupapi.log* file and the corresponding entries in the Registry shows the last time the USB device was connected to the system. To associate this time with the actual time, the Registry key SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones, present in the Software hive is checked.

F. Find if files were opened or copied to USB

We look for entries corresponding to the USB drive letter in the ntuser.dat hive of the identified user (if not deleted explicitly) under the following keys (see figure 4.3, 4.4.):

`Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs.`
`Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU`
`Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU`

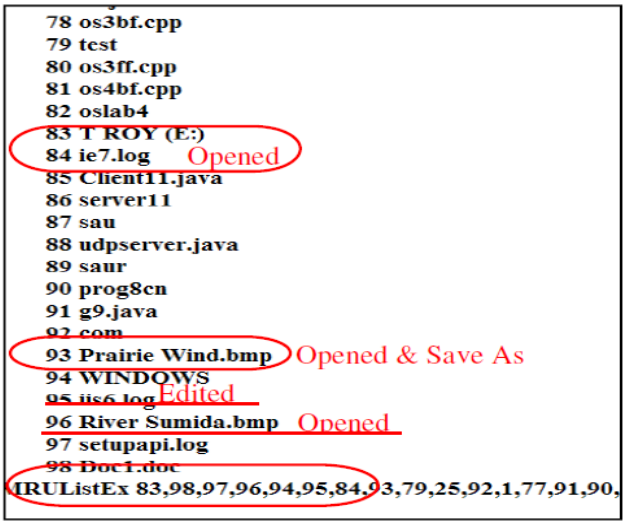

Figure 4.3: An excerpt of the result showing values of RecentDocs Registry key of the identified user.

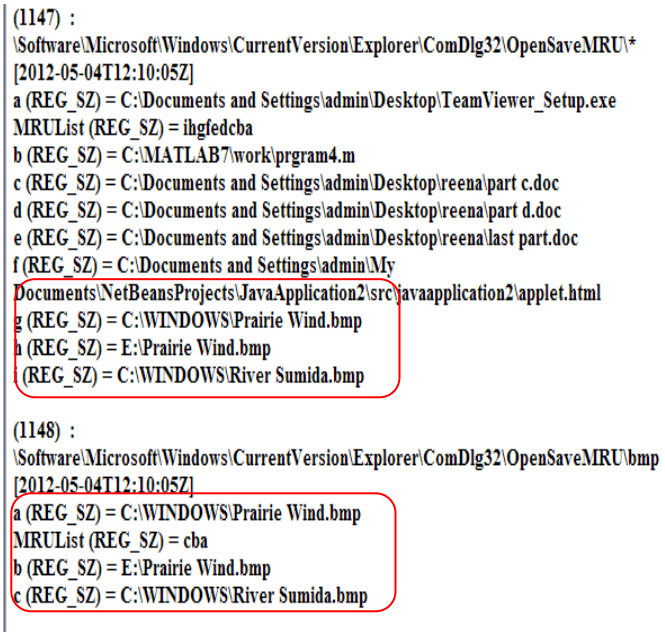

Figure 4.4: An excerpt of the result showing values of OpenSaveMRU Registry key of the identified user.

From the figure 4.4, it is evident that the files *Prairie Wind.bmp* was opened from the system (C:\) then saved to USB (E:\) while *River Sumida.bmp* was opened only. This fact is derived from the entry in OpenSaveMRU. Also the files *ie7.log* and *ii6.log* was accessed through explorer, found from entry in RecentDocs (figure 4.3). However, the files that were copied or moved to USB using the context menu option, has no entry is made in the Registry.

G. Analysis of the file MAC times

When we access (i.e., copy, move, open or edit) a file, their MAC (modified, accessed, created) times gets updated. The MAC times of the files suspected of being copied is checked and compared to the time obtained through Registry analysis. It was observed from MAC times of files in the system that, access times are changed for the files in question and is same as the time the USB device was connected. Hence, it can be concluded that these files were copied.

File name	Operation Performed	Date Modified	Date Created	Date Accessed
Feather Texture.bmp	Copy > Paste	No change	No change	5/4/2012 11:50 AM
Gone Fishing.bmp	Send-to Option	No change	No change	5/4/2012 11:55 AM
Green Stone.bmp	Move (Cut >Paste)	File Not found		
ie7beta2.log	Rename	No change	No change	5/4/2012 11:55 AM
ii6.log	Edit	No change	No change	5/4/2012 11:56 AM
ie7.log	Open(double click)	No change	No change	5/4/2012 11:55 AM
Prairie Wind.bmp	Save through Save-As	No change	No change	5/4/2012 11:59 AM
River Sumida.bmp	Open through explorer	No change	No change	5/4/2012 12:01 AM

Table 5: Change in MAC times of files present in the system.

H. Analysis of the confiscated USB device

Upon knowing the user profile through which the USB device was connected, a case may be build up against the owner of that profile. After confiscating the USB device from that person, a thorough analysis of device's file system is done to find the copied files. We found the files in the USB device and their MAC times were compared to the original files.

Operation Performed	Date Modified	Date Created	Date Accessed
Copy > Paste	Same as old	New, time of copy	Same as date Created
Send-to Option	Same as old	New, time of copy	Same as date Created
Move (Cut >Paste)	Same as old	Same as Old	New, time of move
Rename	Same as old	New, time of copy	Same as date Created
Edit	Time Modified	New, time of copy	Same as date time Modified
Open (double click)	Same as old	New, time of copy	Access Time
Save through Save-As	New, time of save	New, time of save	New, time of save
Open through explorer	Same as old	New, time of copy	Same as date Created

Table 6: Comparison of MAC times of files found in USB with the original files present in the system.

Hence, from *setupapi.log* file, we found the time the USB device (Unique ID- 200402033009D8D25377&0, ParentID Prefix- 7&251e0f1a&0) was connected to the system. The last write times of the corresponding registry keys matched this time. This device was mapped to E: drive and has been connected to the system by User 1. The entries in RecentDocs and OpenSaveMRU key establish the fact that those files were saved E: drive. Finally by analyzing the MAC times of files, details of files copied is found.

The table 7 below summarizes the steps followed in investigating this case and table 8 lists how to track the specific file transfers.

Sl. No.	Action Performed	Subject of Analysis
1	Capture registry files from the system	–
2	Find the time the USB device was connected to the system	registry hive SYSTEM
3	Check the first time the USB device was connected	Setupapi.log
4	Find the drive letter to which the device was mounted, match with the time obtained in step 2 and step 3	registry hive SYSTEM
5	Find through which user profile the USB drive was connected to the system	registry hive ntuser.dat
6	Find if files were opened or saved through explorer, match path with the drive letter obtained in step 4	registry hive ntuser.dat
7	Find if the value `NtfsDisableLastAccessUpdate` is enabled; MAC time of files are checked if value is set to 1	registry hive SYSTEM

Sl. No.	Action Performed	Subject of Analysis
8	Find if files were copied by other modes	MAC time of files present in the system
9	Check for deleted files in the system	Unallocated space of the system
10	Once it is established that files were copied, case is build up against the person using that identified user profile and his USB device is confiscated for analysis	
11	Check for files in USB device, if found compare with original file	MAC time of files in USB
12	If files not found in the USB device, look in deleted space	Unallocated space of USB device
13	Establish file copy by comparing the hash values of original files present in the system and files obtained from the USB	

Table 7: Steps followed in examining file copy to an USB.

Operation Performed	Tracking Method / Footprint Present In
Copy > Paste	File System analysis of system and USB, MAC times compared
Send-to Option	File System analysis of system and USB, MAC times compared
Move (Cut >Paste)	File System analysis of system and USB, MAC times compared
Rename	File System analysis of system and USB, MAC times compared
Edit	RecentDocs Registry key
Open(double click)	RecentDocs Registry key
Save through Save-As	RecentDocs Registry key
Open through explorer	OpenSaveMRU Registry key

Table 8: Tracking specific file transfers

V. Conclusion And Future Scope

Forensic investigations play a significant role in today's working and legal environments, and thus it should be carefully considered. The evidence provided in the registry is the most significant source of any investigation. The actions performed on the computer gives the examiner an insight of the system. Thus, a careful analysis of the Windows system Registry from a forensic point of view is the need of the hour and a hot area of research in the present scenario.

This paper has gathered and verified the existing knowledge about the registry hive files. We also attempted to exhibit the importance of registry analysis by demonstrating how it can help an investigator to progress in a case of tracking data transfer from a system to a USB external device. This work is focused on the examination and generation of registry keys of Windows XP systems only, and can be extended further for the examination of registry files in Windows Vista, Windows 7 and other later versions.

Hence, through the above study it is revealed that through the detailed analysis of the registry hive files, activities of a system user can be traced. Hence registry analysis should be carried as an integral part of digital forensic investigation process.

References

[1] Carvey, H., *The Windows registry as a forensic resource*, Digital Investigation, vol. 2(3), pp. 201–205, Elsevier 2005.

[2] Chang, K., Kim, G., Kim, K. and Kim, W., *Initial Case Analysis Using Windows Registry in Computer Forensics*, Future Generation Communication and Networking, Volume 1, 6-8 Dec. 2007 Page(s):564 – 569. [Online] DOI: 10.1109/FGCN.2007.151 [Accessed 02/11/2011].

[3] Dashora, K., Tomar, D. S. and Rana, J. L., *A Practical Approach for Evidence Gathering in Windows Environment*, International Journal of Computer Applications, Volume 5(10), August 2010.

[4] Farmer, D. J., *A Forensic Analysis of Windows Registry*, Available online from http://forensicfocus.com/downloads/windows-registry-quick-reference.pdf, 2007.

[5] Farmer, D. J., *A Windows Registry Quick Reference: for the Everyday Examiner*, Available online from http://eptuners.com/forensics/contents/A_Forensic_Examination_of_the_Windows_Registry.pdf, 2009.

[6] Kim, Y. and Hong, D., *Windows Registry and Hiding Suspects' Secret in Registry*, In the Proceedings of the 2008 International Conference on Information Security and Assurance, IEEE 2008.

[7] Manchanda, M., Manchanda, V., Gupta, V., Bisht, M., *Forensic Investigation of Window Registry*, International Transactions in Applied Sciences, Volume 2(1), pp. 11-21, January 2010.

[8] Morgan, T., D., *The Windows NT Registry File Format (Version-0.4)*, Available online from http://www.sentinelchicken.com/data/TheWindowsNTRegistryFileFormat.pdf, June 2009.

[9] *Norton Cyber Crime report, 2011*, Symantec Corporation, Available online from http://us.norton.com/content/en/us/home_homeoffice/html/cybercrimereport/, [accessed April 2012].

[10] Russinovich, M. E. and Solomon, D. A., *Management Mechanisms* in *Windows Internals Covering Windows Server 2008 and Windows Vista, (Fifth Edition)*, Microsoft Press, 2009.

[11] *Windows Registry information for advanced users*, Microsoft Support, Available online from http://support.microsoft.com/kb/256986, [accessed April 2012].

[12] *WinHex, , WinHex: Computer Forensics & Data Recovery Software, Hex Editor & Disk Editor*, Available online from http://www.x-ways.net/winhex/

[13] Wong, L. W., *Forensic Analysis of the Windows Registry*. Forensic Focus. Available online from http://www.forensicfocus.com/index.php?name=Content&pid=73&page=1, 2007.