Design and Implementation of WIN-RM (Windows Registry Monitor) Forensic Tool

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Satish Reddy Mallepally

Spring 2011

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. Long-zhuang Li
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

In Windows Operating system, Registry is core component and it contains significant information which is useful for a forensic analyst. It is a repository of the central database in a hierarchal fashion which stores information like configuration of system, applications installed, hardware devices added and user credentials. One such area where rich information can be found is Windows Registry.

Registry is a form of binary data structure which stores values that are considered to be primary replacement of configuration and initialization INI files used in previous windows version systems. Registry can be said as critical resource for digital forensic investigations.

The project focuses on creating a tool that monitors Registry hives and extracts information based on the predefined data. The tool also detects changes made to subtrees, keys and subkeys and their values. A detailed report is showed on the enhanced Registry monitor screen. The projects core idea and focus initially was towards the development of a tool which mainly concentrates on the changes made to the Registry.
TABLE OF CONTENTS

Abstract ........................................................................................................................................... ii

Table of Contents ............................................................................................................................ iii

List of Figures ................................................................................................................................... vi

List of Tables .................................................................................................................................... xi

1. Background and Rationale ........................................................................................................... 1
   1.1 Windows Registry .................................................................................................................. 3
      1.1.1 Structure of Registry .................................................................................................... 3
      1.1.2 Subtree ....................................................................................................................... 4
      1.1.3 Key .......................................................................................................................... 5
      1.1.4 Subkey ...................................................................................................................... 5
      1.1.5 Entry ........................................................................................................................ 6
      1.1.6 Hive Files .................................................................................................................. 6
   1.2 Registry Editor ...................................................................................................................... 10
   1.3 Registry Variability .............................................................................................................. 11
   1.4 Use of Registry .................................................................................................................... 12
      1.4.1 Setup ....................................................................................................................... 12
      1.4.2 Recognizer ............................................................................................................... 12
      1.4.3 Kernel ...................................................................................................................... 12
      1.4.4 Device Drivers ........................................................................................................... 12
   1.5 Regedit and Reg ................................................................................................................. 13

2. Narrative ................................................................................................................................... 14
2.1 Time Stamp Structure ................................................................. 15
2.2 Time Zone Information ............................................................... 16
2.3 Autorun Locations ................................................................. 17
2.4 MRU Lists .............................................................................. 19
2.5 User Assist Keys ................................................................. 20
2.6 Wireless Networks ................................................................. 21
2.7 USB Devices ........................................................................ 25
2.8 LAN Computers ..................................................................... 28
2.9 Mounted Devices ................................................................... 28
2.10 Data Stored in Internet ...................................................... 29

3. Registry Forensics Tools.............................................................. 33
   3.1 Advance Registry Optimizing Tool .......................................... 33
   3.2 Registry Registrar Manager .................................................. 37
   3.3 RegRipper Forensic Tool ...................................................... 40
   3.4 RegLookUp ........................................................................ 41
   3.5 FTK Registry Viewer .......................................................... 42

4. Design System .......................................................................... 47
   4.1 WIN-RM Forensic Tool ....................................................... 47
   4.2 Functionalities of Designed Tool ......................................... 48
   4.3 User Interfacer ................................................................... 48

5. Testing and Evaluation .............................................................. 58
   5.1 Test Case 1: Selecting Hives ............................................... 58
5.2 Test Case 2: Deletion of Keys and Subkeys .............................................. 59
5.3 Test Case 3: Inserting a New Key .............................................................. 60
5.4 Test Case 4: Changing the Data Value ...................................................... 61
5.5 Test Case 5: Viewing the Inserted Key Using WIN-RM Tool ................. 62
5.6 Test Case 6: WIN-RM Tool Detecting the Deleted Key ......................... 62
5.7 Test Case 7: Detecting Change in Data Value Using WIN-RM Tool ....... 63
5.8 Comparison Table .................................................................................. 64
6. Conclusion ............................................................................................... 66
7. Bibliography ............................................................................................. 67
8. Appendix A ............................................................................................... 69
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Structure of Registry</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Subtrees</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Registry Editor</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>Last Write Time</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Time Zone Information</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>System Startup Programs</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>Current User Startup Programs</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Most Recently Used Lists</td>
<td>20</td>
</tr>
<tr>
<td>2.6</td>
<td>Asst Keys</td>
<td>21</td>
</tr>
<tr>
<td>2.7</td>
<td>Network Connections</td>
<td>22</td>
</tr>
<tr>
<td>2.8</td>
<td>Network Interfaces</td>
<td>23</td>
</tr>
<tr>
<td>2.9</td>
<td>Network Profiles</td>
<td>24</td>
</tr>
<tr>
<td>2.10</td>
<td>Network Settings</td>
<td>25</td>
</tr>
<tr>
<td>2.11</td>
<td>USB</td>
<td>26</td>
</tr>
<tr>
<td>2.12</td>
<td>USB Device List</td>
<td>27</td>
</tr>
<tr>
<td>2.13</td>
<td>USB Storage</td>
<td>27</td>
</tr>
<tr>
<td>2.14</td>
<td>Mounted Devices</td>
<td>29</td>
</tr>
<tr>
<td>2.15</td>
<td>Internet Explorer</td>
<td>30</td>
</tr>
<tr>
<td>2.16</td>
<td>Main Subkey</td>
<td>31</td>
</tr>
<tr>
<td>2.17</td>
<td>Typed URL’s</td>
<td>31</td>
</tr>
<tr>
<td>2.18</td>
<td>Internet Explorer Download Directory</td>
<td>32</td>
</tr>
</tbody>
</table>
Figure 4.8 Selection of Hives ................................................................. 54
Figure 4.9 Hives to Monitor ............................................................... 55
Figure 4.10 Start Monitoring ............................................................. 55
Figure 4.11 Registry Report ............................................................... 56
Figure 4.12 Stop Monitoring ............................................................. 57
Figure 5.1 Hives Selection ................................................................. 59
Figure 5.2 Deleting the Key and Subkeys .......................................... 60
Figure 5.3 Inserting a New Key ........................................................ 61
Figure 5.4 Changing the Data Value ................................................ 62
Figure 5.5 WIN-RM Displaying the Occurred Change ...................... 62
Figure 5.6 WIN-RM Detecting the Deleted Entry ......................... 63
Figure 5.7 WIN-RM Detecting Change in Data Value .................... 64
Figure 5.8 Comparison of Different Tools with the Proposed Tool ...... 60
LIST OF TABLES

Table 1.1 Registry Hives and Supporting File .................................................................6

Table 1.2 Registry Data Types...........................................................................................8
1. BACKGROUND AND RATIONALE

When Microsoft first introduced DOS (Disk Operating System), it has its own configuration files config and autoexe which are used to control the settings of operating system and hardware. In DOS, one application does not affect other application as its settings are private and mostly confined to itself. Later Microsoft introduced Windows 3.0, in which new files of type INI called initialization files have been introduced. Next version of windows is Windows 3.1 which concentrates mainly on two basic technologies. The first technology is used for Object Linking and Embedding (OLE) and second technology used is the Drag and Drop. The Drag and Drop technique requires correct, up-to-date information of the specific program or the location.

In order to store and access the information from newly introduced files Windows 3.1 is provided with a registration database called “Registry”. There are many advantages with Registry when compared to INI files. Also Microsoft introduced Windows NT which competes with the UNIX servers and workstations. Registry supports multiple users to use the same workspaces, take turns to share a single computer and cooperate on the sharing of network wide configuration. To achieve these goals Microsoft came with more capable and easy Registry managing technique called NT Registry.

Discovery of the Registry put an end to the struggle with INI files. A few months later, the Windows Registry was featured in a slightly different form, in Windows NT 3.5. Windows NT was initially designed to be used only in networking appliances such as servers and large workstations. Windows NT 3.5 was adapted for use in a multi-user environment, and its features are slightly more complex, but equally fragile Registry.
Throughout the subsequent desktop versions like Windows 98 and Windows Millennium the Registry did not go through too many changes, from Windows 2000 until this new Windows 7 there are slight changes, except that everything is the same in all versions. This prompted the appearance of Registry cleaning and repairing software, which is very helpful in repairing the fragments of Registry whenever necessary.

If Registry is constructed as a database and if it offers features like atomic updates which is a strong feature in the databases, then the integrity of the system will be improved highly. One process change is done after another. That is at a given point of time no two processes updates its Register values at same time, then data is maintained consistently. Sometimes while performing this, changes are made to INI files, if such case occurs then a race condition occurs then data values provided will be inconsistent and does not match with updated values.

Windows 7 and Windows Vista use the commit-abort semantics, to provide updates to the Registry in a transaction manner with atomicity. Thus updating of the Registry values should be transactional and each value of Registry should posses a time-stamped updation. This will be used in any forensic investigation by a forensic analyst and is a major clue when the updation has happened.

Previously similar tools have been developed but not as robust as the proposed WIN-RM tool. None of the tools like RegLookUp, Adv Optimizer Reg and FTK Registry Viewer previously have an option for selecting required hives for monitoring, time saving, detailed report. Previous tools only dealt with one or more hives, have no option to select the hives and there results are cumbersome to view. Proposed tool overcomes all limitations of previously developed tools.
1.1 Windows Registry

Windows Registry is a hierarchical database which stores information about the system configuration for single or multiple users, applications installed in the system and services that run on system. The applications and operating system itself use this Registry to store values or data about user and configuration of the current system. The Registers are available all time until the system is running. All programs keep data persistently in Registers so as to refer for any future use.

A forensic investigator discovers lots of information from the Registries such as when software has been installed, when it is updated and deleted. Also a forensic analyst can check which hardware devices are connected and at what time these devices are connected and disconnected. The investigator has a log of internet files viewed and the analyst can use that history for any further investigation.

Apart from that Registry also contains valuable data and settings for users of the software that has been installed, for instance if any changes made to control panel settings, file associations, system policies or installed software the changes are reflected and stored in software. This Registry mechanism is same in file system, here sysfs and procfs are used. Although the information presented by both differs.

1.1.1 Structure of the Registry

Structure of the Registry is of nested kind which is called as subtrees, keys and subkeys which are similar to that of files and folders in the system. Registry entries are used to store actual data, which are lowest level Registries. There are series of continuous links between each entry until the lowest entry. Each entry has a unique name [Carvey Harlan 2007] with which individual entries are identified. Whenever necessary the entries
are referenced by their entire path and with their names. Figure 1.1 shows the structure of the Registry with the subtrees, keys and subkeys. It also shows names of all the registry keys, their datatype and the value associated with that particular key.

Figure 1.1 Structure of Registry

1.1.2 Subtree

Subtrees are the primary division or the basic root of the Registry. Windows server 2008 is divided into five subtrees. Subtrees does not possess any of the configuration data or values, subtrees further contain keys, subkeys and lower level entries.

The Five subtrees of Windows server 2008 are

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

HKEY_CURRENT_CONFIG

Figure 1.2 shows five subtrees. Windows Editor shown in figure 1.2 has two panels in screen, one shows links between subtrees, keys and subkeys, [Carvey Harlan 2007] other shows name, type of data and value associated with the key.

![Figure 1.2 Subtrees](image)

### 1.1.3 Key

Keys are next level of subtrees, which has at least one subkey, for example the hardware keys. Sometimes subtrees will have no subkeys.

### 1.1.4 Subkey

Subkeys are next level of abstraction of the tree flow. This stores the entries and other subkeys
1.1.5 Entry

Registry entries are the lowest level elements. In Registry Editor entries appear in the right pane of the window. Each entry is a combination of entry name, datatype and value of [Carvey Harlan 2007] Registry entry. Here in these entries configuration of operating system and some application programs are stored. Entries are entirely different from subtrees, keys and subkeys.

1.1.6 Hive Files

Hives are the files in which the permanent Registry values are stored. The location of this hive list subkey is in
HKLM\SYSTEM\CurrentControlSet\Control

And files are saved in systemroot\System32\Config; it gets updated after each and every login. The four of five keys are stored in HKEY_LOCAL_MACHINE and one key in HKEY_USERS [Carvey Harlan 2007]

Table 1.1 shows Registry hives and supporting files.
HKEY_LOCAL_MACHINE has the four different subkeys and supporting files are given in the table.

**Table 1.1 Registry Hives Supporting Files**

<table>
<thead>
<tr>
<th>Registry hive</th>
<th>Supporting files</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKEY_CURRENT_CONFIG</td>
<td>System, System.alt, System.log, System.sav</td>
</tr>
<tr>
<td>HKEY_CURRENT_USER</td>
<td>Ntuser.dat, Ntuser.dat.log</td>
</tr>
<tr>
<td>HKEY_LOCAL_MACHINE\SAM</td>
<td>Sam, Sam.log, Sam.sav</td>
</tr>
</tbody>
</table>
Following are hives contained in the Windows Registry

- **HKEY_CLASSES_ROOT**:  
  **HKEY_CLASSES_ROOT** also abbreviated as the HKCR, this stores information of all applications that are registered. Flexible design allows either use or machine registration COM objects.

- **HKEY_CURRENT_USER**:  
  **HKEY_CURRENT_USER** is also abbreviated as HKCU. This saves settings specific to the users who logged in currently. Each of the user’s settings is stored in their respective files known as NTUSER.DAT and USRCLASS.DAT, which reside in their own Documents and Settings Subfolder

- **HKEY_USERS**:  
  **HKEY_USERS** is abbreviated as HKU, and contains subkeys which are stored in HKEY_CURRENT_USER when each user is logged into the machine, but actual user hives are stored when the user currently logs in.

- **HKEY_CURRENT_CONFIG**:
**HKEY_CURRENT_CONFIG** is abbreviated as HKCC. HKCC stores information or data collected at runtime. Data collected is not permanently stored in the disk, in fact this information is acquired at the runtime. If anyone wants to view current profile setting of the user who is using this, just simply follow the path

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\HardwareProfiles \Current.
```

- **HKEY_DYN_DATA**

  **HKEY_DYN_DATA** hive is only present in Windows 95, 98 and Millennium. This stores dynamic status information of plug and play devices and is not stored in hard drives and is only stored in memory, information may be changed depending upon addition or removal of devices connected to system.

  Table 1.2 shows name of the value used and datatype of that value and description given for that particular datatypes.

**Table 1.2 Registry Data Types [MS 2008]**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Value</td>
<td>REG_BINARY</td>
<td>Raw binary data. Most hardware component information is stored as binary data and is displayed Registry Editor in hexadecimal format</td>
</tr>
<tr>
<td>DWORD</td>
<td>REG_DWORD</td>
<td>Data represented by a number that is</td>
</tr>
<tr>
<td>Value</td>
<td>REG_SZ</td>
<td>4 bytes (a 32-bit integer). Many parameters for device drivers and services are this type and are displayed in Registry Editor in binary, hexadecimal or decimal format. Related values are DWORD_LITLLE_ENDIAN (least significant byte is at the lowest address) and REG_DWORD_BIG_ENDIAN (least significant byte is at the highest address)</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Expandable string value</td>
<td>REG_EXPAND_SZ</td>
<td>A variable length data string. This data type includes variables that are resolved when a program or service uses a data</td>
</tr>
<tr>
<td>Multi String value</td>
<td>REG_MULTI_SZ</td>
<td>A multiple string. Values that contain multiple values in a form that people can read are generally this type. Entries are separated by commas, spaces and other marks</td>
</tr>
<tr>
<td>String Value</td>
<td>REG_SZ</td>
<td>A fixed length string</td>
</tr>
<tr>
<td>Binary Value</td>
<td>REGRESOURCELIST</td>
<td>A series of nested arrays that is designed to store a resource list that is used by a hardware device driver or one of the physical devices it controls. The data is detected and written in the Resource Map tree by the system and is displayed in Registry Editor in Hexadecimal format as a binary value.</td>
</tr>
<tr>
<td>Binary Value</td>
<td>REGRESOURCEREQUIREMENTSLIST</td>
<td>A series of nested arrays that is designed to store a device drivers list of possible hardware resources the driver of one of the physical devices it controls can use. The system writes a subset of the list in ResourceMap tree. This data is detected by system and is displayed in Registry Editor in Hexadecimal format a Binary value.</td>
</tr>
</tbody>
</table>

### 1.2 Registry Editor

It is easy for user or for system administrator to have a smooth navigation between subtrees and subkeys. Registry Editor provides a fragile abstraction layer between the Registries in the system. User has to be very careful while dealing with
Registry Editor because if register values are corrupted, whole system may go wrong and may stop working. Most of the administrators many times use the Application program interface (API’s) to change Registries whenever administrators want to add, delete or modify values in Registries. Fastest way to open editor is go to Run and then type regedit.exe or type regedit in run and then click ok. Figure 1.2 shows the Registry Editor.

![Registry Editor](image)

Figure 1.2 Registry Editor

1.3 Registry Variability

All Registry values with in a particular system will be different. A system may have different software’s and hardware’s installed in it. Values of Registries present in system are very specific to the computer and its configuration. So whenever user system is logged off and after restarting it, Registry values may change. And they are stored in different location than the previous ones. To add,
retrieve, delete or to make any changes to the Registries, use standard Win32 Application Program Interface (API’s).

1.4 Use of Registry Data

Programs typically use Win32 application program interface to get data values from the Registry. The program specifies path and name and in turn the API returns value of the Register.

1.4.1 Setup

Setup program adds configuration data to Registries. For instance new information is added whenever a user wants to install some software or add a new hardware.

1.4.2 Recognizer

When a computer is started, each time the job of recognizer is to place hardware configuration data in Registry. The data includes hardware devices connected to system. Ntdetect.com and kernel Ntoskrnl.exe programs are used.

1.4.3 Kernel

At the time of the system startup, kernel reveals information about itself to Registry. Also it pulls information from Registry, for example information of the device drivers and...Etc.

1.4.4 Device Drivers

Device drivers send and receive information about load parameters and available configuration data from Registry. Also device drivers must have a log of information containing system resources it is using.
1.5 Regedit.exe and Reg.exe

Regedit.exe is a Registry Editor and Reg.exe is a command line tool which is used to restore and backup Subkeys, Keys or the entire Subtree by keeping the parts of the Registry as files in system or any removable media.

Reg.exe which is command line tool is used to modify local registries and Regedit.exe is used to modify both remote and local registries.
2. NARRATIVE

Windows Registry has vital information which is very useful for a forensic analyst. There are some evidences found which are important to deal with any kind of case, to obtain clues from collected evidence, mainly such as time zone information, when the operating system was installed and system turn on and off time logs, and time when external hardware devices connected.

Forensic investigator while examining, will prepare a detailed document of all activities that have been taken place during the examination in parallel. Most researchers have analyzed and examined Registries; most crucial thing is interpreting Registry information. A small mistake in the interpretation can give rise to a big trouble in analysis. Only experienced analyst can tackle critical issues in process of investigation.

Computer forensics consists of four different phases, Collection, Examining, Analysis and Reporting. First phase is collection phase is searching, collecting and documentation of all evidences obtained. Second one, examination process helps to find evidence visible, describes the origin and significance. Examination should be documented so that state of evidence is proved totally. Documentation helps to understand the hidden and obscured evidence.

Third phase is the analysis phase, is to observe the product of examination for its significant output. Usually examinations and analysis are done by two different forensic specialists, but in some cases both are done by single analyst. In many cases the forensic examination is done by forensic analyst and then the analysis part is taken care by the investigator. Fourth phase is the preparation of the final report, is statement or report that
summarizes the examination process and results. The reports must be preserved for future testimony or for a defense against further examination to validate actions.

Some tools such as RegRipper and RegLookUp are available for the forensic investigators for extracting the evidences from Registries. Further this data is examined and analyzed to produce a detailed report.

2.1 Time Stamp Structure

Every Registry key contains a unique value associated with it called “LastWrite” time, which is almost all equal to last modified time of a particular file. The associated value is represented as a FILENAME structure and gives information when a particular Registry was last modified. The LastWrite time is generally updated at the time of the creation of the Registry key. Also it is updated whenever the Registry key is modified, accessed or deleted. Main drawback in this is one can obtain only the LastWrite time of the Registry key but not the Registry value.

After obtaining LastWrite time value of the Registry key forensic analyst has a clear picture of date and time of event occurred. Also knowing LastWrite time of the Registry key itself does not solve the problem, as it is still difficult to determine what value was actually changed. By using log of LastWrite time of the Registry key and other sources of information gathered, one can come to some general conclusion of the event happened. Figure 2.1 shows the LastWrite time of the software Navicat premium
2.2 Time Zone Information

Time Zone Information is referred as critical and most important evidence, which is assumed to be pretty consistent. There are certain values in the key which are very helpful in determining day light saving time and different time zones. The acquired will be again converted to UTC standard timestamps to local timestamps.
Figure 2.2 Time Zone Information

Figure 2.2 shows the time zone information of the current system and the daylight saving time information. When a particular time is changed or whenever user changes time zones, changes in time reflect and keys are going to be changed. According to those values associated to that particular key also changes. Figure 2.2 shows name of the key at the left column in right side panel, datatype is in middle and value associated with that key is at the right corner.

2.3 Autorun Locations

Autorun Locations are locations in Registry where applications and programs are launched at system startup process. As soon as system boots location of the programs are decided in the Registries. Figure 2.3 shows locations in Registry where applications and programs are launched at system startup process. It is system start up process at which it shows what data it contains and what key name and its datatypes.
Figure 2.4 shows when user start up, it shows current user log, when user changes or logs into other account. When user first starts system name of the key, datatype of key and value is present in system Registry.
2.4 MRU Lists

MRU is abbreviation for the “Most Recently Used Lists” stores entries which are caused by some actions performed by user. Here numerous number of MRU lists are present in registry keys. Obtained MRU lists are stored in one log file for any future reference.

Figure 2.5 shows most recently used keys and their subkeys with their values and the datatypes. “Most Recently Used Lists” stores entries. Here numerous number of MRU lists are present in Registry keys. With the help of this MRU a forensic analyst can make out the actions performed by user.
2.5 User Assist Key

User Assist Key has two or more subkeys, which has very large names in hex format or called Globally Unique Identifier (GUIDs). Under the GUIDs, it has a subkey known as count. This count subkey contains values of the objects that user had used or accessed on system.

Figure 2.6 shows user assist key, which has many subkeys and value associated is very large in hexadecimal format called GUID’s.
2.6 Wireless Networks

Ethernet card picks the range in which it has maximum signal strength from wireless access points. Wireless Networks are identified by SSID or service set identifier. When any user connects to hotspot or a network, the SSID is logged in network connection. [MS 2008]

Figure 2.7 shows network connections to which system is connected. It shows all the network names of the wireless networks. Also it shows the Tcpip connections established as illustrated in the figure 2.7.
Figure 2.7 Network Connections

Figure 2.8 shows interfaces that are in the parameters of the network connection.

Figure 2.8 shows names of the interfaces, to which datatype it belongs and the value associated with that key.
Figure 2.8 Network Interfaces

Figure 2.9 illustrates all the network profile presents all keys which describe networks till to date to which the system is connected. It has subkeys and the values associated with it in the given profiles.
Figure 2.9 Network Profiles

Figure 2.10 shows all the network settings of the system. It has all the information about the server name and DHCP connections and when the system is connected to which wireless network and their respective keys.
2.7 USB Devices

Whenever any external device is plugged into the USB, all drivers of the device are initially queried and the information of the device is stored in Registry. For example all thumb drives, cameras and ….etc. Also this key contains the subkey that describes the device ID, vendor ID and Product ID of all the devices that are connected to the system.

Figure 2.11 shows all the USB devices that are connected to the local system and it has the entire configuration and the names of the USB devices connected to it. And keys have the unique values in it. Figure 2.12 illustrates all the keys present in the USB.
Figure 2.11 USB

Figure 2.12 lists all the USB devices list which are available in USB list store and it has the information about the thumb drives or any hardware connected to the system with the date and time. These time stamp values are very much important for the forensic analyst to come to a conclusion and provide the evidence in the court of law.
Figure 2.12 USB Devices List

Figure 2.13 shows all the USB devices and their information regarding the name, the configuration, the drivers attached to it.

Figure 2.13 USB Storage
2.8 **LAN Computers**

In the Windows family, especially the window XP implements the tool called Network Place, which maps the network [MS 2008]. This allows the computer to locate others computers with in the LAN and it records all computer names connected to this particular network and the date and time when devices are connected or disconnected from the network.

2.9 **Mounted Devices**

Registry stores all the information of the mounted volumes in the NTFS file system. It is possible to view each individual drive connected to the system. The information is in the form of binary data in \DosDevices\x, specifies each value connected to that system.

Figure 2.14 illustrates all the mounted devices connected to the system. Clearly the name of the mounted device appears, their datatypes and data value associated with it.
2.14 Mounted Devices

2.10 Data stored in Internet

Internet explorer has three subkeys in which majority of data is stored for investigation or if any crime or misuse is done with the system.

Figure 2.15 shows all the data in internet explorer key.
There are three subkeys, first subkey contains the Main, which has user settings, it stores useful information such as the form settings start page and search pages. There is one interesting and useful data in this key pertaining to the user passwords called “FormSuggest”. If this value is “Yes” then its good and it says that the particular system is protected from the password savings [MS 2008]. If it is “No” then it indicates that each and every time the user logins to the account the passwords are been saved. Figure 2.16 illustrates all the data of the search bars and settings of the system. In the left side panel of the screenshot one can observe for the form suggest value is “No”. It clearly indicates that the system is protected from any password misuse.

Figure 2.15 Internet Explorer
Figure 2.16 Main Subkey [Joan 2010]

The figure 2.17 shows all the URL’s viewed by the user.

Figure 2.17 Typed URL’s [Joan 2010]
The next subkey is used to store the information of the recently downloaded files from online like the internet sources. Figure 2.18 shows all the downloads files which are recently downloaded from the internet.

Figure 2.18 Internet Explorer Download Directory
3. Registry Forensic Tools

Most of Registry forensic tools find changes in the Registries, like if the value is changed or removed or created. Registry forensics tools stores all the data about when and where data is changed, with time and date of that particular change made. There are many open source software in internet where initial forensic analyst can start with [MS 2008]. There are many closed source software tools also. The commercial tools available are somewhat more critical to understand and these tools are more furnished than the tools available freely online.

The main job of these tools is to edit the values in the Registry. Some tools tries to repair the damage done to the hives, and some tools are efficient enough to clean all the hive files and dat files.

3.1 Advance Registry Optimizing tool

Advanced Registry Optimizing tool is an open source tool that scans all the Registry hives and gives a detailed report of all the errors in the hive files. This tool is capable of scanning whole registries, taking back up of all Registries [Joan 2010]

Figure 3.1 shows the Registry optimizing tool and its options to select which ever task a user want to perform.
Figure 3.1 Scan Registry

Figure 3.2 shows the scanning process of the registry

Figure 3.2 Process of Scanning
Figure 3.3 illustrates the system status and all the categories in which the errors are. For example in figure 3.3 the applications, filetypes, histories in which the scanning took place can be viewed.

Figure 3.3 Detailed Report of the Error

Figure 3.4 shows the current user hives errors and in each and every location it shows the 64 or 32 bit. And it is very clear to see the hives location.
Figure 3.4 Current User Registry Errors

Figure 3.5 illustrates all the errors in the local machine, it has the path towards the subkeys and the value can be seen at that subkey.

Figure 3.5 Errors in the Local Machine
3.2 Registry Registrar Manager

For the forensic analysts this tool is very useful to collect important data relating to changes in the Registries. It is used to store repositories and maintain data not only in the desktops computers but also the remote computers. It mainly used for the restoring the defragmented Registries, adding additional description to the Registry keys and values. It sets the access restrictions and monitors Registry access.

Figure 3.6 illustrates the Registry monitor tool. In this, the options are seen clearly as to select the registry hives and can monitor it for any changes. On the right side of the panel there is a last write time also which is very much useful in the forensic investigation point of view.

![Figure 3.6 Registrar Registry Manager](image)
In the figure 3.7 one can clearly see the options like to select the Registry hives, and to select to look at which part of the hive, that is in key names or values or data.

**Figure 3.7 Advanced Registry Search**

Figure 3.8 lists all the operation performed on the hives, including the current activities to monitor. It is easy for the forensic analyst to select the particular hive, a particular process to monitor and a specified subkey.
In the figure 3.9 one can observe the actions performed, on which particular process the action is performed, and location of the hive, value it represents.
3.3 RegRipper Forensic Registry Tool

This is one of the fastest and the easiest tool used by the forensic analyst. The main reason why it is simple to understand is, this uses the Perl. This tool is mainly discovered and works intelligently for Windows 2000, XP and 2003 hive files. One of the most important things in this tool is that it does not touch the hives directly, instead it extracts the files using some tools like FTK Imager and it then checks for the misuse or for any modifications. This tools outputs the results in a CVS file format and it makes use of the plug-ins to access the Registry keys. Ripping of hive file and report file together is possible.

Figure 3.10 shows the Registry Ripper Tool Ripping Hive and Report file
3.4 RegLookUp Registry Monitor

This is a command-line utility which allows the users to read the Registries and query it. Most of these are implemented on Windows NT and higher versions. It reads all the Registry values and in a detailed standadized format all the values are seen. Figure 3.11 shows the Registry start up locations, path of the subkeys and the time.

![Figure 3.11 RegLookUp](image-url)
3.5  FTK Registry Viewer

This tool plays a vital role in the forensic point of view, the investigator finds this one handy in dealing critical cases and with the evidences found one can have a clue that what has happened. This tool provides detailed and useful information about the passwords and the usernames, the information that cannot be accessed by the regedit can be accessed by this tool. Also it accesses the encrypted protected data for any important [Joan 2010] Registry information. The Registry values are displayed in a full view. The final report can be obtained by selecting the views of the keys.

Figure 3.12 shows the domains of the account, users, names of the administrators and the value associated with it.

![Figure 3.12 Access Data Registry Viewer](image)
Figure 3.13 shows how Security analyzer manager, makes sure that all the operations performed on the hives are recorded and secured.

Figure 3.13 Security Analyzer Manager (SAM)

Registry Viewer has the tendency to access the Registries protected area, which stores vital data not accessible in Windows regedit for forensic analysis.

Figure 3.14 shows the options of the panel, in the left bottom all the last written time values are obtained.
Figure 3.14 Advance Search Criteria

After scanning the required hive files the data is displayed and report is displayed to see all the changes in the hives. Figure 3.15 shows how the analysis is done and then creates the results report.
Figure 3.15 Report of Viewer

Figure 3.16 shows the results of the Registry. It contains the information like file, type and the data.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>File1</td>
<td>REG_SZ</td>
<td>C:\Documents and Settings\User\Desktop\Search Words.rtf</td>
</tr>
<tr>
<td>File2</td>
<td>REG_SZ</td>
<td>C:\cases\data\test\case.dat</td>
</tr>
<tr>
<td>File3</td>
<td>REG_SZ</td>
<td>K:\New\kismet5.txt</td>
</tr>
<tr>
<td>File4</td>
<td>REG_SZ</td>
<td>K:\New\madwifi.txt</td>
</tr>
</tbody>
</table>

Figure 3.16 Registry Report Information
4. **Designed System**

There are many forensic tools in market that does provide a variety of registry data and a detailed report to get some clues how an event is occurred. But each tool has its own limitations, such as one tool provides all data of the modified keys, and removed keys and doesn’t provide the data and values of the changed keys. Some tools like RegLookUp and RegRipper only provide changes in the values do not provide at what time does changes happened. In many cases the tool does not provide the entire information of the registries like when and at what time keys are created, when keys are modified, and when these keys are deleted. The information related to the hive files are not in a single place. There comes the idea to develop this tool.

The designed tool WIN-RM, has overcome the limitations of the tool discussed. Unlike other registry monitoring tool, which monitor all the hives irrespective of the incident, the developed tool has the option to select the hive in which a particular incident is taken or a change in the local system. For example user knows that a change in the HKEY_USER and in the control panel, this particular tool allows choosing a particular hive in which the Registries should monitor.

4.1 **Proposed WIN-RM (Windows Registry) Forensics Tool**

The main objective of the tool developed is to allow forensic investigator to view changes made in the Registries. The tool implements new feature that has never been implemented in any of the available tools. This particular tool has been developed by C # programming language.
4.2 Functionalities of Designed Tool

The main advantage of the tool is that if at all one knows that in any company or an organization someone is modifying or deleting some important information, then this tool can be installed in that suspected system and it starts monitoring the system, then after that seeing the data available from a tool the forensic analyst can easily decide at what time the change has happened, and at which place of the file and what information is modified with the date and time.

- The Registry hives are selected by using the settings
- The drop down menu in the Registry hive contains all hives in it.
- Selection of the hives is done according to the incident response
- After selecting the hives, start monitoring any changes in the hives
- Stop monitoring the hives
- A detailed list will be displayed showing the changes made to the keys or a message saying that a key is created or removed
- Bottom of the window panel have all the buttons for the keys as mentioned
  - Created Keys
  - Created Values
  - Removed Keys
  - Removed Values
  - Changed values

It becomes very easy for the forensic analyst to examine the data if all the data is at one place.
Program Flow Diagrams

Figure 4.1 shows the program flow of the designed tool.

Figure 4.1 Program flow

Figure 4.2 shows the MainForm class structure and its attributes
Figure 4.2 MainForm

Figure 4.3 shows the RegMon class and its attributes

Figure 4.3 RegMon
Figure 4.4 shows the setting hives class and its attributes

**Figure 4.4 Setting Hives**

Figure 4.5 shows the RegKey class and its attributes

**Figure 4.5 RegKey**
4.3 **User Interfaces**

Whenever the tool is executed the enhanced Registry monitor screen pops up and it has the options like monitor, settings and help. The Registry hives are selected by using the settings then the drop down menu in the Registry hive contains all hives in it. The hives present in it are

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG

Selection of the hives is done according to the incident response. After selecting the hives, start monitoring any changes in the hives, if the changes occur, stop monitoring the hives. A detailed list will be displayed showing the changes made to the keys or a message saying that a key is created or removed[Joan 2010]. The detailed list of all the entries, which makes ease for any forensic investigator. It becomes very easy for the forensic analyst to examine the data if all the data is at one place.

Figure 4.6 shows the view of the home page of the tool, which has all the options as seen in the figure.
Figure 4.6 Enhanced Registry Monitor

Figure 4.7 shows the menu in the settings to select the hives present in it.

Figure 4.7 Registry Hives
Figure 4.8 shows all the hives present in the Registries. It has the option to select one of those or all of those or some of those and then just click ok.

![Figure 4.8 Selection of Hives](image)

Figure 4.8 Selection of Hives

Figure 4.9 shows that after selecting the HKEY_CURRENT_USER it shows that only that particular hive is going to monitor.
Figure 4.9 Hives to Monitor

Figure 4.10 shows that the start of the monitoring is going to take place.

Figure 4.10 Start Monitoring
Figure 4.11 shows the Registry report, detailed list will be displayed showing the changes made to the keys or a message saying that a key is created or removed. It becomes very easy for the forensic analyst to examine the data if all the data is at one place.

Figure 4.11 Registry Report

The figures 4.12 illustrate the stop monitor option after all the scanning is done.
Figure 4.12 Stop monitoring
5. Testing and Evaluation

By these test cases one can easily evaluate the tool has a peculiar characteristic of collecting the data for more than a day. By this one can say that the rating for this tool will be high enough, though it has some limitation as the other tools have. The tool does not keep the database stored, so in future implementation this tool should be integrated with database including atomicity part.

5.1 Test Case 1: Hives Selection

Whenever the tool is executed the enhanced Registry monitor screen pops up and it has the options like monitor, settings and help. The Registry hives are selected by using the settings then the drop down menu in the Registry hive contains all hives in it. The hives present in it are Selection of the hives is done according to the incident response. After selecting the hives, start monitoring any changes in the hives, if the changes occur, stop monitoring the hives.

Figure 5.1 shows how the selection of the hives is done according to the need of the investigator.
5.2 Test Case 2: Deletion of Keys and Subkeys

After selecting the hives, then go into that particular hive and then try do delete any of the key or a subkey. Figure 5.2 shows the HKEY_CURRENT_USER hive and deleting the key.
5.3 Test Case 3: Inserting a new Key

While inserting a new key into the hive just go to the system Registry editor and open the specified location where the insertion to be made and then the change is viewed in the tool instantaneously.

Figure 5.3 shows that a new key is being inserted.
5.4 Test Case 4: Changing the Data Value

While the system hives are being monitored, the change can be done in the particular hive. Figure 5.4 shows how a data value is being changed.
5.5 Test Case 5: Viewing the Inserted Key in Win RM Tool

The inserted key is spontaneously showed in the tool box. The figure 5.5 shows that the tool detected the change and is updated.

![Win RM Displaying the Occurred Change](image)

Figure 5.5 Win RM Displaying the Occurred Change

5.6 Test Case Win RM Detecting the Deleted Entry

The designed tool Win RM detects the deleted entry. Figure 5.6 shows the deleted entry from the hive. All the changes are seen in the middle with the black letters white background. The deleted entries are shown in the bottom below removed keys.
5.7 Test Case 7: Detecting Change in the Data Value using WIN-RM Tool

Whenever the change in the data value of the Registry is made, immediately the designed tool shows that change and also it stores the changes under the changed values. Figure 5.7 shows the change in the data value detected by the tool.
Figure 5.7 WIN-RM Detecting the Change in Data Value

Comparison of Different Tools with Designed Tool

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hive Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date &amp; Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key Created</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Removed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tool features</td>
<td>Adv. Reg. Optimizing Tool</td>
<td>RegRipper</td>
<td>RegLookUp</td>
<td>FTK Registry Viewer</td>
<td>WIRECHAD E</td>
<td>WIN-RM Tool (proposed)</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------------------</td>
<td>------------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>Value Removed</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Value Changed</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Instant changes</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Duration Support</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Export to excel sheet</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.8 Comparison of Different Tool with the Proposed Tool**
6. Conclusion

As windows operating system is more popular in homes and businesses, it is a basic need of computer forensic experts to know the complexity and understand the magic of Windows Registry. The Windows Registry is a rich source where a forensic analyst can find more clues and evidences to come to a decision. Potential evidence which resides in registries can help forensic analyst about what actions might be occurred on the machine. Whole idea of this particular study is to find the rich repository of the changes made to the system, where the examination of data is done to find the cause or the reasons for that cause. Some tools are discussed with their features.

The designed tool WIN-RM is having its own features, which are important for any forensic investigator because it makes their work easy. The designed tool performs a scans in selected hives. It gives the detailed report of all the changes made, deleted and inserted keys. And a good user interface makes for a forensic investigator to easily understand what is going on.

The enhancements for the tool will be the integration of the database of the collected data. This integration helps to store all the changes made on each and everyday and in future if anyone wants to refer it.
7. BIBLIOGRAPHY

21 Dec 2004. 8 Apr 2007

[Davies Peter 2007] "Forensic Analysis of the Windows Registry."


[Adrian 2009], "Why Choose the Opera Internet Suite," Opera wiki. 2007. 13 Apr


[Ruth Supree 2010] (WIRECHADE) Windows Registry Change Detection Forensics, Graduate Project, TAMUCC, 2010
Appendix A

KeyReg Class

Encapsulates a RegistryKey definition, that is a key-name with several values and several sub-keys. For comparison purposes, two hashes are created for the key’s values and for the key’s sub-keys, this allows us to determine in a slightly faster way which areas in the structure are different, instead of comparing all keys one-to-one (which would take quadratic time, or worst…). Despite of being quick in comparison, the generation of this structure (using fromRegistryKey) may take a few seconds to be completed.

KeyValue

Stores information about a registry key’s value, that is, it’s type, its name and its value.

RegMon

Creates a registry change event using Windows API and waits for changes on the specified hive. An initial cache of the hive is made on the “regkey” object, the “rkey” is the RegistryKey object pointing to the hive. When a change is detected, the structure from the hive is read again and compared against the stored structure (on regKey). Changes are notified via the RegChanged event.

Program Code

// Program.cs
using System;
using System.Windows.Forms;
RegMon
internal sealed class Program //This is the first Class called and its calling Main Function
{
    [STAThread]
    private static void Main(string[] args)
    {
        Application.EnableVisualStyles();

        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new MainForm()); // main form class is being created and called by the Application.Run(new MainForm())
    }
}

// in main class, in the constructor of that class.....initialize component() function is called

namespace RegMon
{
    partial class Settings_Hives
    {
        /// <summary>
        /// Designer variable used to keep track of non-visual components.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Disposes resources used by the form.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose(disposing);
        }
    }
}

/// <summary>
/// This method is required for Windows Forms designer support.
/// Do not change the method contents inside the source code editor. The Forms designer might
/// not be able to load this method if it was changed manually.
/// </summary>
private void InitializeComponent() // this function is initializing different variables and parameters like creating menus tabs and anything else
    this.label1 = new System.Windows.Forms.Label();
    this.HiveList = new System.Windows.Forms.CheckedListBox();
    this.SuspendLayout();
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(12, 8);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(220, 18);
    this.label1.TabIndex = 3;
    this.label1.Text = "Select Registry Hive(s) to Monitor:";
    //
    // HiveList
    //
    this.HiveList.FormattingEnabled = true;
    this.HiveList.Location = new System.Drawing.Point(12, 29);
    this.HiveList.Name = "HiveList";
    this.HiveList.Size = new System.Drawing.Size(333, 166);
    this.HiveList.TabIndex = 2;
    //
    // OkButton
    //
    this.OkButton.Location = new System.Drawing.Point(12, 298);
    this.OkButton.Name = "OkButton";
    this.OkButton.Size = new System.Drawing.Size(125, 23);
    this.OkButton.TabIndex = 1;
    this.OkButton.Text = "OK";
    this.OkButton.UseVisualStyleBackColor = true;
    this.OkButton.Click += new System.EventHandler(this.OkButton_Click);
    //
    // CancelButton
    //
    this.CancelButton.Location = new System.Drawing.Point(12, 329);
    this.CancelButton.Name = "CancelButton";
    this.CancelButton.Size = new System.Drawing.Size(125, 23);
    this.CancelButton.TabIndex = 0;
    this.CancelButton.Text = "Cancel";
    this.CancelButton.UseVisualStyleBackColor = true;
    this.CancelButton.Click += new System.EventHandler(this.CancelButton_Click);
    //
    // SuspendLayout()
    //
    this.SuspendLayout();
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(12, 8);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(220, 18);
    this.label1.TabIndex = 3;
    this.label1.Text = "Select Registry Hive(s) to Monitor:";
    //
    // HiveList
    //
    this.HiveList.FormattingEnabled = true;
    this.HiveList.Location = new System.Drawing.Point(12, 29);
    this.HiveList.Name = "HiveList";
    this.HiveList.Size = new System.Drawing.Size(333, 166);
    this.HiveList.TabIndex = 2;
    //
    // OkButton
    //
    this.OkButton.Location = new System.Drawing.Point(12, 298);
    this.OkButton.Name = "OkButton";
    this.OkButton.Size = new System.Drawing.Size(125, 23);
    this.OkButton.TabIndex = 1;
    this.OkButton.Text = "OK";
    this.OkButton.UseVisualStyleBackColor = true;
    this.OkButton.Click += new System.EventHandler(this.OkButton_Click);
    //
    // CancelButton
    //
    this.CancelButton.Location = new System.Drawing.Point(12, 329);
    this.CancelButton.Name = "CancelButton";
    this.CancelButton.Size = new System.Drawing.Size(125, 23);
    this.CancelButton.TabIndex = 0;
    this.CancelButton.Text = "Cancel";
    this.CancelButton.UseVisualStyleBackColor = true;
    this.CancelButton.Click += new System.EventHandler(this.CancelButton_Click);
this.OkButton.Location = new System.Drawing.Point(192, 201);
this.OkButton.Name = "OkButton";
this.OkButton.Size = new System.Drawing.Size(73, 23);
this.OkButton.TabIndex = 4;
this.OkButton.Text = "OK";
this.OkButton.UseVisualStyleBackColor = true;
this.OkButton.Click += new System.EventHandler(this.OkButtonClick);
  
// CancelButton
  
this.CancelButton.Location = new System.Drawing.Point(272, 201);
this.CancelButton.Name = "CancelButton";
this.CancelButton.Size = new System.Drawing.Size(73, 23);
this.CancelButton.TabIndex = 5;
this.CancelButton.Text = "Cancel";
this.CancelButton.UseVisualStyleBackColor = true;
this.CancelButton.Click += new System.EventHandler(this.CancelButtonClick);
  
// Settings_Hives
  
this.ClientSize = new System.Drawing.Size(354, 230);
this.Controls.Add(this.CancelButton);
this.Controls.Add(this.OkButton);
this.Controls.Add(this.label1);
this.Controls.Add(this.HiveList);
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
this.MaximizeBox = false;
this.MinimizeBox = false;
this.Name = "Settings_Hives";
this.Text = "Select Registry Hives | Settings";
this.Load += new System.EventHandler(this.Settings_HivesLoad);
this.ResumeLayout(false);
this.PerformLayout();
}
private System.Windows.Forms.CheckedListBox HiveList;
private System.Windows.Forms.Label label1;

//After creating the GUI form when we select the start monitor this function gets called
void StartMonitoringToolStripMenuItemClick(object sender, EventArgs e)
{
    if (SelectedHives.Count == 0)
    {
        MessageBox.Show ("Please first select which registry hives to monitor.", "General Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    this.Height = 494;
    Log.Items.Clear (); //simply clears all the logs
    this.startMonitoringToolStripMenuItem.Enabled = false;
    this.stopMonitoringToolStripMenuItem.Enabled = true;
    foreach (String hive in SelectedHives)
    {
        RegMon rm = new RegMon (hive);
        rm.RegChanged += new EventHandler(OnRegChanged); // calling the function OnRegChanged function this is the event function which get triggred whenever there is any change in the registry
        rm.Error += new System.IO.ErrorEventHandler(OnError);
        rm.Start();
        Monitors.Add (new object[] { hive, rm });
        log ("Monitoring of Hive " + hive + " Started.");
    }
    // function OnRegChanged
private void OnRegChanged(object sender, EventArgs e)
{
    if (InvokeRequired)
    {
        BeginInvoke(new EventHandler(OnRegChanged),
                    new object[] { sender, e });
        return;
    }

    RegMon rm = (RegMon)sender;
    RegKey newRegKey = RegKey.fromRegistryKey(rm.rkey);
    ArrayList results = new ArrayList();
    rm.regKey.compareTo(newRegKey, results);
    string now = DateTime.Now.ToString();
    StreamWriter sw;
    foreach (RegChangeInfo inf in results)
    {
        switch (inf.type)
        {
            case RegChangeInfo.VALUE_CREATED:
                dataGridView1.Rows.Add(new object[] { now, inf.p_key, inf.p_name, inf.p_type, inf.p_value });
                sw = new StreamWriter("VALUE_CREATED.csv", true);
                sw.WriteLine(now + "","" + inf.p_key + "," + inf.p_name + "," + inf.p_type + "," + inf.p_value);
                sw.Flush();
                sw.Close();
                log ("New value Created.");
                break;

            case RegChangeInfo.VALUE_REMOVED:
                dataGridView2.Rows.Add(new object[] { now, inf.p_key, inf.p_name, inf.p_type, inf.p_value });
                sw = new StreamWriter("VALUE_REMOVED.csv", true);
                break;
        }
    }
}
sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name + "," + inf.p_type + "," + inf.p_value);
sw.Flush();
sw.Close();
log ("Value removed.");
break;

case RegChangeInfo.VALUE_CHANGED:
dataGridView4.Rows.Add (new object[] { now, inf.p_key, inf.p_name, inf.p_type, inf.p_value, inf.n_value });
sw = new StreamWriter
("VALUE_CHANGED.csv", true);
sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name + "," + inf.p_type + "," + inf.p_value + "," + inf.n_value);
sw.Flush();
sw.Close();
log ("Value has changed.");
break;

case RegChangeInfo.KEY_CREATED:
dataGridView3.Rows.Add (new object[] { now, inf.p_key, inf.p_name });
sw = new StreamWriter
("KEY_CREATED.csv", true);
sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name);
sw.Flush();
sw.Close();
log ("Key created.");
break;

case RegChangeInfo.KEY_REMOVED:
dataGridView5.Rows.Add (new object[] { now, inf.p_key, inf.p_name });
sw = new StreamWriter
("KEY_REMOVED.csv", true);
sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name);
sw.Flush();
sw.Close();
log ("Key removed.");
break;
private void InitRegistryKey(string name) {
    string[] nameParts = name.Split('\\');
    switch (nameParts[0]) {
        case "HKEY_CLASSES_ROOT":
        case "HKCR":
            _registryHive = HKEY_CLASSES_ROOT;
            regKey = RegKey.fromRegistryKey (rkey = Registry.ClassesRoot);
            break;

        case "HKEY_CURRENT_USER":
        case "HKCU":
            _registryHive = HKEY_CURRENT_USER;
            regKey = RegKey.fromRegistryKey (rkey = Registry.CurrentUser);
            break;

        case "HKEY_LOCAL_MACHINE":
        case "HKLM":
            _registryHive = HKEY_LOCAL_MACHINE;
            regKey = RegKey.fromRegistryKey (rkey = Registry.LocalMachine);
            break;

        case "HKEY_USERS":
            _registryHive = HKEY_USERS;
            regKey = RegKey.fromRegistryKey (rkey = Registry.Users);
            break;

        case "HKEY_CURRENT_CONFIG":
            _registryHive = HKEY_CURRENT_CONFIG;
            regKey = RegKey.fromRegistryKey (rkey = Registry.CurrentConfig);
            break;

        default:
            _registryHive = IntPtr.Zero;
            throw new ArgumentException("The registry hive " + nameParts[0] + " is not supported", "value");
    }
}
_registrySubName = String.Join("\\", nameParts, 1, nameParts.Length - 1);
}

// function called by Initregistrykey()
public static RegKey fromRegistryKey (RegistryKey rk)
{
    RegKey regKey = new RegKey();

    regKey.updateHash (regKey.name = rk.Name);

    foreach (string name in rk.GetValueNames ())
    {
        try
        {
            object o = rk.GetValue (name);
            RegistryValueKind k = rk.GetValueKind (name);

            KeyValue kv;

            if (k == RegistryValueKind.Binary)
            {
                kv = new KeyValue (name, k.ToString(), Convert.ToBase64String ((byte[])o));
            }
            else
            {
                byte[] o = System.Text.UTF8Encoding.ASCII.GetBytes (o.ToString ());
                kv = new KeyValue (name, k.ToString(), Convert.ToBase64String (o));
            }

            regKey.updateHash2 (kv.name);
            regKey.updateHash2 (kv.type);
            regKey.updateHash2 (kv.value);

            regKey.values.Add (kv);
        }
        catch (Exception e)
        {
        }
    }

    // updating of the hashes
    public static Int64 getHash (string value)
    {
        Int64 hash = 0;

        int sign = 1;
        byte pv = 175;

        foreach (byte v in System.Text.ASCIIEncoding.ASCII.GetBytes (value))
        {
            hash += sign * (v + pv) * (pv - v);
            sign = sign > 0 ? -1 : 1;
        }
    }
}
pv = v;
}
return hash;
}

// Updates the internal hash.
private void updateHash (string value)
{
    khash += getHash (value);
}

private void updateHash2 (string value)
{
    Int64 x = getHash (value);
    khash += x;
    vhash += x;
}

// after that it reaches rm.start()
public void Start()
{
    if (_disposed)
        throw new ObjectDisposedException(null, "This instance is already disposed");

    lock (_threadLock)
    {
        if (!IsMonitoring)
        {
            _eventTerminate.Reset();
            _thread = new Thread(new ThreadStart(MonitorThread));
            _thread.IsBackground = true;
            _thread.Start();
        }
    }
}

// it comes to monitor thread
private void MonitorThread()
{
    try
    {
        ThreadLoop();
    }
    catch (Exception e)
    {
        OnError(e);
    }
    _thread = null;
}
// from there it goes to Monitor Thread
private void ThreadLoop()
{
    IntPtr registryKey;
    int result = RegOpenKeyEx(_registryHive,
    _registrySubName, 0, STANDARD_RIGHTS_READ | KEY_QUERY_VALUE |
    KEY_NOTIFY,
            out registryKey);
    if (result != 0)
        throw new Win32Exception(result);

    DateTime lastTime = DateTime.MinValue;

    try
    {
        AutoResetEvent _eventNotify = new
        AutoResetEvent(true);
        WaitHandle[] waitHandles = new WaitHandle[]
        {_eventNotify, _eventTerminate};
        while (!_eventTerminate.WaitOne(0, true))
        {
            result =
            RegNotifyChangeKeyValue(registryKey, true, _regFilter,
            _eventNotify.Handle, true);
            if (result != 0) throw new
            Win32Exception(result);

            if (WaitHandle.WaitAny(waitHandles) ==
            0)
            {
                DateTime curTime = DateTime.Now;
                if (curTime.Subtract
                (lastTime).TotalSeconds >= 1)
                {
                    lastTime = curTime;
                    OnRegChanged();
                }
            }
        }
    }
    finally
    {
        if (registryKey != IntPtr.Zero)
        {
            RegCloseKey(registryKey);
        }
    }
} // calling function

[DllImport("advapi32.dll", SetLastError = true)]
private static extern int RegNotifyChangeKeyValue(IntPtr hKey, bool bWatchSubtree, RegChangeNotifyFilter dwNotifyFilter, IntPtr hEvent, bool fAsynchronous);

// so whenever there is a change in the registry it is raising the event OnRegChanged()
protected virtual void OnRegChanged()
{
    EventHandler handler = RegChanged;
    if (handler != null) handler(this, null);
}

// as we have set the handler using
rm.RegChanged = newEventHandler(OnRegChanged), finally the function in mainframe gets called
// function OnRegChanged
private void OnRegChanged(object sender, EventArgs e)
{
    if (InvokeRequired)
    {
        BeginInvoke(new EventHandler(OnRegChanged),
        new object[] { sender, e });
        return;
    }

    RegMon rm = (RegMon)sender;
    RegKey newRegKey = RegKey.fromRegistryKey(rm.rkey);

    ArrayList results = new ArrayList();

    rm.regKey.compareTo(newRegKey, results);// compares the current register value to the previous values

    string now = DateTime.Now.ToString();
    StreamWriter sw;

    foreach (RegChangeInfo inf in results)
    {
        switch (inf.type)
        {
            case RegChangeInfo.VALUE_CREATED:
                dataGridView1.Rows.Add(new object[] { now, inf.p_key, inf.p_name, inf.p_type, inf.p_value });
                sw = new StreamWriter("VALUE_CREATED.csv", true);
        }
sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name + "," + inf.p_type + "," + inf.p_value);
sw.Flush();
sw.Close();
log ("New value Created."); break;

case RegChangeInfo.VALUE_REMOVED:
    dataGridView2.Rows.Add (new object[] { now, inf.p_key, inf.p_name, inf.p_type, inf.p_value });
    sw = new StreamWriter ("VALUE_REMOVED.csv", true);
    sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name + "," + inf.p_type + "," + inf.p_value);
    sw.Flush();
    sw.Close();
    log ("Value removed."); break;

case RegChangeInfo.VALUE_CHANGED:
    dataGridView4.Rows.Add (new object[] { now, inf.p_key, inf.p_name, inf.p_type, inf.p_value, inf.n_value });
    sw = new StreamWriter ("VALUE_CHANGED.csv", true);
    sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name + "," + inf.p_type + "," + inf.p_value + "," + inf.n_value);
    sw.Flush();
    sw.Close();
    log ("Value has changed."); break;

case RegChangeInfo.KEY_CREATED:
    dataGridView3.Rows.Add (new object[] { now, inf.p_key, inf.p_name });
    sw = new StreamWriter ("KEY_CREATED.csv", true);
    sw.WriteLine (now + "," + inf.p_key + "," + inf.p_name);
    sw.Flush();
    sw.Close();
log ("Key created.");
break;

case RegChangeInfo.KEY_REMOVED:
    dataGridView5.Rows.Add (new object[] { now, inf.p_key, inf.p_name });

    sw = new StreamWriter
       ("KEY_REMOVED.csv", true);
    sw.WriteLine (now + "," + 
       inf.p_key + "," + inf.p_name);
    sw.Flush();
    sw.Close();
    log ("Key removed.");
    break;
}