5-2012

Exploring the Uses of ShellBag Data within the Windows 7 Registry.

Daniel A. Duncan
East Tennessee State University

Exploring the Uses of ShellBag Data within the Windows 7 Registry

Thesis submitted in partial fulfillment of Honors

By

Daniel Duncan
The Honors College
University Honors Scholars Program
East Tennessee State University

10 April 2012

Dr. Michael R. Lehrfeld, Faculty Mentor

Mr. Patrick Cronin, Faculty Reader

Dr. Ronald Zucker, Faculty Reader

DOCUMENT INFORMATION

Author(s): Daniel Duncan, duncanda@goldmail.etsu.edu

Classification: Public

Keywords: Windows ShellBag, Digital Forensics, Computer Forensics, BAG parser, Windows Explorer, Shell, Bags, BagMRU, Window settings, RecentDocs, StreamMRU

VERSION

Version	Date	Comments
1.0	8/29/2011	Initial version based on research done in Spring 2011.
1.1	10/20/2011	Added initial table of contents, overview of program, and Abstract
1.2	12/7/2011	Added Introduction
1.3	12/13/2011	Added Shell BAG Analysis
1.4	1/27/2012	Updated Shell BAG Analysis
1.5	1/28/2012	Added sources. updated Glossary, added Document Information, updated Acknowledgements
1.6	1/29/2012	Added Algorithm section
1.7	2/14/2012	Updated Algorithm section
1.8	2/28/2012	Updated Syntax and Structure section
1.9	2/29/2012	Added the Conclusions section
1.10	3/1/2012	Added Appendices, Overall proofreading changes
2.0	3/7/2012	Applied initial proofreading comments from mentor
2.1	3/18/2012	Applied second proofreading comments from mentor
3.0	3/28/2012	Updated Algorithm section
3.1	4/2/2012	Updated Algorithm section, added Source Code and Program Results sections

ABSTRACT

Digital forensic examiners are faced with the task of recreating a user's actions for auditing purposes. ShellBag data from the registry is critical to the reproduction of these actions in a Microsoft Windows 7 operating system, because ShellBag data contains a listing of folders and files contained within a specific folder. Once an understanding of the structure of ShellBag data in a Windows 7 operating system is reached, this data can be parsed to create a timeline of user actions on a given machine.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Professor Ronald Zucker and Mr. Patrick Cronin for their assistance in the proofreading and preparing this manuscript. In addition, special thanks to Dr. Michael Lehrfeld whose knowledge and experience in this field gave the idea to take on this research for the advancement of the field of digital forensics. Thanks to Billy Overton for his help in creating the proof-of-concept script for this project. Last but not least, thanks to my fiancée, Jamie, for her encouragement during this project.

INTRODUCTION

Overview

Think about the process involved when a police officer responds to a crime report and examines a computer at the scene of the crime that supposedly contains child pornography. ShellBags provide an enhanced tracking system of user actions within Windows Explorer due to the fact that the Windows Registry uses ShellBags to store records of all folders that have ever been accessed and when those folders were accessed.

The Windows Registry is the storage place for all user settings and pointers to data stored on a computer's hard drive. According to some recent research by Zhu, Gladyshev, and James (2009) on the structure of the ShellBag section of the Windows Registry, the information gleaned from this part of the Registry is the key. ShellBag data is used by Windows for the arrangement of files and folders within the Windows Explorer view. However, they in turn contain snapshots of contents within folders when these folders are viewed at different screen resolutions, which hold critical information about what files have been created, modified, and deleted. As a result, this data is critical to recreating user events to use as evidence in criminal prosecution.

ShellBag data is not readily available to the user because of its location and its format within the Windows Registry. To access this information, a program that parses the data within ShellBags is needed. To address this need, a script has been written in Windows PowerShell to illustrate this parsing process.

What is Computer Forensics?

To understand why this research has taken place, it is important to understand the field to which it applies. Forensics is defined as "the process of using scientific knowledge for collecting, analyzing, and presenting evidence to the courts." Computer forensics is then defined as "the discipline that combines elements of law and computer science to collect and analyze data from computer systems, networks, wireless communications, and storage devices in a way that is admissible as evidence in a court of law," (US-CERT, 2008). This research will serve the digital forensics field by giving practitioners an understanding of how ShellBag data is collected and analyzed.

What is the Registry?

The study of ShellBags is dependent upon a clear understanding of the Windows Registry. According to the *Microsoft Computer Dictionary,* the registry is defined as "A central hierarchical database used in Microsoft Windows 98, Windows CE, Windows NT, and Windows 2000 used to store information that is necessary to configure the system for one or more users, applications and hardware devices," (Microsoft Support, 2008). Windows references the information found in the Registry frequently during system uptime to successfully execute programs, open documents, and create, modify, or delete files and folders in the file system.

Before the Registry, Windows used text-based .ini files to hold system configurations for the user.

In summary, the registry is a database that stores references to files, settings, applications used during the time that a user is logged on. In addition, a clear understanding of the registry structure is required before analyzing ShellBags.

Registry Structure

The Windows Registry is divided into hives, which function as tree-like structures which contain keys, which contain sub-keys, which in turn contain values (also referred to as properties). There are 5 main hives within the Registry, all of which are explained in Table 1. Each hive contains settings that relate either to users or to the computer itself.

Windows Registry Hives	
Name	**Function**
HKEY_CURRENT_USER	Contains settings for the current user such as screen color, background, and folder storage structure
HKEY_USERS	Contains profiles for every user on the machine
HKEY_LOCAL_MACHINE	Contains information about the computer's configuration
HKEY_CLASSES_ROOT	Contains information that opens ensures a program's correct execution
HKEY_CURRENT_CONFIG	Contains information about the hardware used by the computer

Table 1: This table lists the five Registry hives and a brief description of the functionality of each hive.

HKEY_CURRENT_USER and HKEY_USERS contain settings that refer to users on the computer. The HKEY_CURRENT_USER hive contains the root of the configuration information for the current logged in user. Settings for the user such as screen color, background, and folder storage structure are kept here. The HKEY_USERS hive contains the actively loaded profiles on the computer for every user that has logged into the machine.

The HKEY_LOCAL_MACHINE, HKEY_CLASSES_ROOT, and HKEY_CURRENT_CONFIG hives contain settings for the computer itself. The HKEY_LOCAL_MACHINE hive contains information about the configuration of the computer itself. The HKEY_CLASSES_ROOT hive holds the information that ensures that the correct program is opened when a file is selected to open using Windows Explorer. This same information can also be found under the HKEY_LOCAL_MACHINE and the HKEY_CURRENT_USER keys. In the HKEY_LOCAL_MACHINE\Software\Classes key, default settings for all users can be found. In the HKEY_CURRENT_USER\Software\Classes key, settings chosen by the individual user are stored and would be applied instead of the default settings. The HKEY_CLASSES_ROOT hive blends these two sources of information. However, settings must be modified under the HKEY_CURRENT_USERS key for custom settings or under the HKEY_LOCAL_MACHINE key for default settings. The last hive, HKEY_CURRENT_CONFIG, contains information about the hardware used by the local computer upon startup (Davies, 2006).

Registry Location

The most common way to view the Registry is to run the regedit.exe program. This program presents a graphical representation of the registry which visibly illustrates the tree-like

structure of the Registry. The information found within the registry can be viewed in more ways than just the regedit.exe GUI, such as viewing the text-based *ntuser.dat* system file in a traditional text editor or by navigating to the Registry from within the Windows PowerShell interface. Registry values are also stored in files on the hard drive. In Windows 7, these files are organized in much of the same way as the in the Windows XP and Vista file systems, which is under the path C:\Windows\System32\config. This directory contains an assortment of files with varying extensions. Files with a .dat or no extension are the basic registry files, which were used in older versions of Windows. Files with any kind of .log extension have traditionally been used to keep record of any changes made to the registry. In Windows 7, these log files have been split up into two parts: .LOG1 and .LOG2. .LOG1 files contain the current list of changes made to the registry, and .LOG2 contains the original snapshot of the registry, thus replacing the traditional .sav files which were used in the past to hold the original snapshot of the registry before any installations or modifications took place. The HKEY_CURRENT_USER values are stored elsewhere in the file system under the path C:\Users\{UserName} and in the file ntuser.dat (Davies, 2006).

Due to permission issues, the user is not allowed to access and edit the registry value files. However, one can navigate to the registry hives in Windows PowerShell by issuing a command to change the directory to one of the aforementioned registry hives. For example, one could access the HKEY_CURRENT_USERS hive by typing the command *cd HKCU:* into the PowerShell command prompt. PowerShell treats the Registry in the same manner as it does the File System, so the user can navigate through the keys like he would normal directories.

This means that the user can view the contents of the keys, and thus analyze the contents of ShellBags.

SHELL BAG ANALYSIS

How Windows Operating Systems Use ShellBags

ShellBag data is used by Windows for the arrangement of files and folders within the Windows Explorer view. It is also used to store both recently and frequently used applications in the Start Menu, as well as data about recently used folders stored anywhere else on the machine. Forensics experts have found that this data provides tracks of deleted files and folders, network paths, and external media once plugged into the system. ShellBag data can give this kind of information because it takes snapshots of the file structure within any mounted media, including removable media. ShellBag data provides consistency in how the user sees windows in Windows Explorer from one session to another (Zhu, Gladyshev & James, 2009).

If the user resizes a window in Windows 7, data about the files and folders within the resized folder are written to sub-keys within the key *HKEY_CURRENT_USER**Software**Microsoft**Windows**Shell**Bags*. It is possible for investigators to recreate folder contents using this data alone, but since the data is stored in binary blobs, it is relatively obscure to computer users with no coding knowledge. A binary blob is "a collection of binary data stored as a single entity," (Wikipedia contributors, 2011). These binary blobs are stored in hexadecimal format, so they must be converted to a readable format, such as ASCII encoded characters. In addition to the format of ShellBags, their locations present a different problem, because this data is stored in six different locations in the Windows 7 Registry. Each

location directly corresponds to one other location. For example, the relationship between *BagMRU* and *Bags* is one where one location, *BagMRU,* houses a list of **M**ost **R**ecently **U**sed folders, and each MRU entry refers to one specific entry in the other location, *Bags,* which holds a complete list of all folders ever accessed (Davies, 2006).

ShellBag Location

In Windows 7, ShellBag data can be found in the *HKCU**Software**Microsoft**Windows**Shell* path in two different locations:

- *Bags*
- *BagMRU*

The *HKCU**Software**Classes**Local Settings**Software**Microsoft**Windows**Shell* path houses ShellBag data in two repositories:

- *Bags*
- *BagMRU*

Finally, two other repositories hold ShellBag information at the *HKCU**Software**Microsoft**Windows**CurrentVersion**Explorer* path:

- *StreamMRU*
- *RecentDocs* (Khatri, 2011).

In previous versions of Windows, there were other paths that held ShellBag data. The *ShellNoRoam* folder under the *HKCU**Software**Microsoft**Windows* path held the settings for

remote folders, while the settings for local folders were held in the *Shell* folder under the same path. After XP, the *ShellNoRoam* folder was removed and replaced with the *Shell* key at the *HKCU**Software**Classes**Local Settings**Software**Microsoft*\Windows path which contains *Bags* and *BagMRU* keys. As one can see, the *HKCU**Software**Microsoft**Windows**Shell* path makes up the latter part of this other path, but since it is under the *Local Settings* key, this proves that the new path holds settings for local folders. The *Bags* and *BagMRU* folders under the *HKCU**Software**Classes**Wow6432Node**Local Settings**Software**Microsoft**Windows**Shell* path also do not exist in Windows 7.

Once the locations of ShellBag data have been established, analysis of this data can begin. Again, due to the obscure nature of the ShellBag format, the structure of a Bag file must be analyzed to provide a way to parse this data and convert it to a readable format.

Syntax & Structure

By itself, the *BagMRU* key traditionally represents the Desktop. This is because the file system recognizes the Desktop as the root folder for everything else. Under this key are only keys that are named 0 or 1. The first 0 is representative of "My Documents," and the first 1 is representative of "My Computer." Under "My Computer", the C: and D: drives are represented by the next 0 and 1, respectively. Each key under *BagMRU* can have a maximum of three different types of properties, as illustrated in Figure 2. The first property is a listing of items that have been most recently accessed arranged in numerical order. The second is called

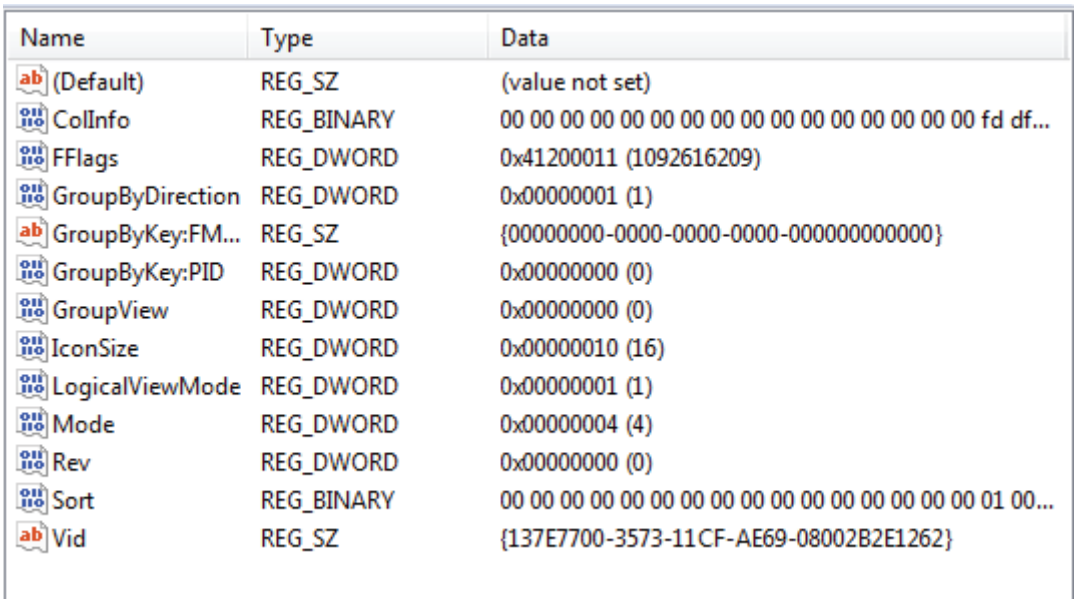
MRUListEx, and it records the sequence of the MRU items. The most recently updated item is found in the first four bytes in the binary form of the entry. The next four bytes represent the second most recently accessed item, and so on. The last property is called *NodeSlot*. It is a reference to a corresponding entry about the same item under the *Bags* folder in the same path. These properties hold data about the creation, modification, and last access times as well as the name of the folder to which the entry refers (Zhu, Gladyshev & James, 2009). This uncovers a particular truth about ShellBags. If the Registry is a database, then *BagMRU* is a table that holds the most recently accessed items. Like any relational database, entries in the *BagMRU* table act as foreign keys to the *Bags* table. This is important because of the relationship between *BagMRU* and *Bags.* As previously mentioned, *BagMRU* holds only the most recently used folders, and the entries in this list contain a property called *NodeSlot* whose value directly correspond to the base-ten numerically named entries in *Bags.*

Name	Type	Data
(Default)	REG_SZ	(value not set)
0	REG_BINARY	e7 02 31 00 00 00 00 00 5c 3d 9a 96 11 00 44 6f 77 6e...
MRUListEx	REG_BINARY	00 00 00 00 ff ff ff ff
NodeSlot	REG_DWORD	0x00000007 (7)

Figure 2: Shown above are the three possible properties found under a *BagMRU* subkey.

The *Bags* key is differently structured than *BagMRU* in that its subkeys' naming system is not binary. Instead, the sub-keys are simply named numerically starting with 0. The *Shell* key found under each numerically named sub-key holds the display settings for a given folder, as

shown in Figure 3. These settings include starting window position, which view mode has been selected (i.e. icons, list, details, tiles, content), and how items have been sorted within the folder (i.e. by Name, Date Modified, Type, Size, etc.). The maximum number of sub-keys under *Bags* is indicated in the *BagMRU Size* value under the *Shell* key (Zhu, Gladyshev & James, 2009).



Name	Type	Data
(Default)	REG_SZ	(value not set)
ColInfo	REG_BINARY	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 fd df...
FFlags	REG_DWORD	0x41200011 (1092616209)
GroupByDirection	REG_DWORD	0x00000001 (1)
GroupByKey:FM...	REG_SZ	{00000000-0000-0000-0000-000000000000}
GroupByKey:PID	REG_DWORD	0x00000000 (0)
GroupView	REG_DWORD	0x00000000 (0)
IconSize	REG_DWORD	0x00000010 (16)
LogicalViewMode	REG_DWORD	0x00000001 (1)
Mode	REG_DWORD	0x00000004 (4)
Rev	REG_DWORD	0x00000000 (0)
Sort	REG_BINARY	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00...
Vid	REG_SZ	{137E7700-3573-11CF-AE69-08002B2E1262}

Figure 3: Illustrated above are the properties found under a typical *Shell* key, including sort method, icon size, and view mode, among others.

It is important to note that ShellBag information is created only after folders are accessed in Windows Explorer. Therefore, folders that may exist on the hard drive but have never been accessed will not have corresponding ShellBag information. The study by Zhu, Gladyshev & James (2009) has shown that this information is not created until the folder that has been opened in Windows Explorer has been closed. The method to how this information is created depends on where folders reside on the hard drive and if they have been accessed

before. The reason location is relevant is because the Desktop is viewed as the root folder in the Registry. The researchers conducted their experiments by comparing snapshots of the Registry in a Windows XP virtual machine before and after a certain action were performed. The results of these experiments are outlined in the Tables 2.1-2.9 below.

Experiment 1				
Location	Action	Prexisting SB Data?	Update	Result
Desktop	Open	True	"MRUListEx" property is updated in the *BagMRU* key to reflect the order in which the most recently opened items are listed first.	The target MRU item is marked as the most recently used item In the *BagMRU* key.
Desktop	Close	True	"MRUListEx" property is updated to say that the folder is the most recently opened folder.	N/A

Table 2.1: In the first experiment, the researchers opened and closed a folder located in the Desktop that had previously been opened and thus had existing corresponding ShellBag data. The results of this experiment showed how the Registry updates the *BagMRU* key is for a folder with preexisting ShellBag data.

Experiment 2				
Location	Action	Prexisting SB Data?	Update	Result
Below Desktop	Open	True	*BagMRU* key is updated first. The position of the target item is updated in the sequence value of the "MRUListEx" property.	Each parent folder is updated and becomes the most recently accessed folder in the list.
Below Desktop	Close	True	Same process as above.	Same process as above.

Table 2.2: In the second experiment, the researchers opened and closed a folder in a directory located in a hierarchical tier below the Desktop that had previously been opened. The results of this experiment also illustrate that the *BagMRU* key is updated in the same sequence for folders in and below the Desktop.

Experiment 3				
Location	Action	Prexisting SB Data?	Update	Result
Desktop	Open	False	The system enumerates the contents of the *BagMRU* key.	No new Bag information is created.

Table 2.3: In the third experiment, the researchers only opened a folder located on the Desktop with no preexisting ShellBag data. The results showed that ShellBag entries are not created when a folder is opened.

Experiment 4				
Location	Action	Prexisting SB Data?	Update	Result
Below Desktop	Open	False	The system enumerates the contents of the *BagMRU* key.	No new Bag information is created, but the target folder's parents' MRU items' position was updated, starting with the *BagMRU* key and ending when no existing items within the MRU key matching the target folder are found.

Table 2.4: In the fourth experiment, the researchers again only opened a folder with no preexisting ShellBag data, but this time, the folder was located below the Desktop. The results for this experiment prove that although no new ShellBag entries are created when a folder is opened, if the folder is located in a directory below the Desktop, the parent folders above the target folder are updated to have entries in the *BagMRU* key's list.

Experiment 5				
Location	Action	Prexisting SB Data?	Update	Result
Desktop	Close	False	The system enumerates the contents of the *BagMRU* key to find matching ShellBag data for the folder, but finds none.	Registry creates a new item in the *BagMRU* key and a new sub-key of *BagMRU* associated with the new folder.

Table 2.5: In the fifth experiment, the researchers closed a folder located on the Desktop that had no preexisting ShellBag information. The results showed that the new MRU item created contains the target folder's name and timestamp information. The system also updated the *BagMRU* key's "MRUListEx" property by moving the newly created key to the most recently used position. Finally, the "Display" property of the target folder was written, and its display settings were written to its Shell sub-key.

Experiment 6				
Location	Action	Prexisting SB Data?	Update	Result
Below Desktop	Close	False	Same as Experiment 5.	Same as Experiment 5. In addition, the folder's ancestors' MRU items' position was marked as the most recent item in the corresponding "MRUListEx" property.

Table 2.6: In the sixth experiment, the researchers repeated the process of Experiment 5, except the folder closed was located below the Desktop. The results showed that before making the newly closed folder the most recently used item, the folder's parents were each sequentially marked as the most recently used item. This is because the user must first theoretically access the parent folders before accessing the target folder.

Experiment 7				
Location	Action	Prexisting SB Data?	Update	Result
Any	Delete	True	None.	The ShellBag information is not deleted, so it remains.

Table 2.7: In the seventh experiment, the researchers deleted a randomly chosen folder from the file system. The results showed that even though the folder was deleted, its ShellBag data

remained in the Registry. This is an especially significant finding for the purposes of digital forensic research.

Experiment 8				
Location	Action	Prexisting SB Data?	Update	Result
Any	Make new folder with the same name as another folder	True	Existing ShellBag information for the pre-existing folder is updated.	The timestamp information within the MRU item remains the same.

Table 2.8: In the eighth experiment, the researchers made a new folder located in the same folder as another folder with the same name. The results showed that the preexisting folder's ShellBag data was updated, and a new entry was not made. This brought another interesting proposition to light. If folders with the same name and path share a spot in the ShellBag data, how can you tell which one is the original one? This can be done by simply comparing the creation timestamp in the MRU item to the creation timestamp of the folder stored in that path. If the two times match, the item may be deemed the original.

Experiment 9				
Location	Action	Prexisting SB Data?	Update	Result
Any	Make new folder, but maximum number of Display keys has been reached	False	The folder's Display key is not created but assigns the "NodeSlot" property under the folder's MRU key to 1 and updates the corresponding Shell key.	There are now two folders associated with this key.

Table 2.9: In the ninth and final experiment, the researchers created a new folder after the operating system had already stored the maximum number of ShellBag keys possible. The results showed that the operating system starts over with the first entry ever created and added a new *NodeSlot* property under the key. This shows that there can be two folders associated with the same ShellBag entry.

Based on experiments conducted, there are three types of updates caused by user actions:

- Updates to the target folder and its ancestors' MRU items' position if these MRU items exist. No new ShellBag entries are created.

- Update both the relevant MRU items' position and the contents of the Shell sub-key under the folder's Display key. New ShellBag entry is created.
- No ShellBag information is involved.

In the case of a forensic examiner, the first and second types of user actions are important because they are what yield results in the ShellBag area. However, due to results in the experiments, one might see a difficulty that can arise in examining ShellBag data. Since the information is updated after different kinds of actions, it can be tricky to determine which item actually caused the change in the data (Zhu, Gladyshev & James, 2009).

To aid in the assembly of a Bag parser, it was important to discover the structure of the bag file itself. Bag entries have the following structure:

- Most entries start with a solitary integer, followed by 3 carry bytes
- Next 4 bytes are Last Modified Date
- Carry 3 bytes
- Next 4 bytes are short DOS name
- Carry 8 bytes
- Next 4 bytes are creation date and erroneous time
- Next 4 bytes are the Last Accessed Date and real Creation time
- Carry 6 bytes
- Followed by long File Name
- Carry 20 bytes, end entry (Hay, 2004).

The *RecentDocs* key can be useful to a forensic examiner to see a list of the last 20 user-accessed items given in the same binary blob format. The key itself gives this listing, as shown below in Figure 4. Even more valuable are the sub-keys of *RecentDocs,* which list the file extensions for the most recently used files and a separate "Folders" sub-key for folders that have been accessed, which is illustrated in Figure 4. Therefore, the contents of each sub-key contain each file that has been recently accessed that has that given file extension. If the item accessed is not a file, folders are given their own sub-key entitled "Folder."

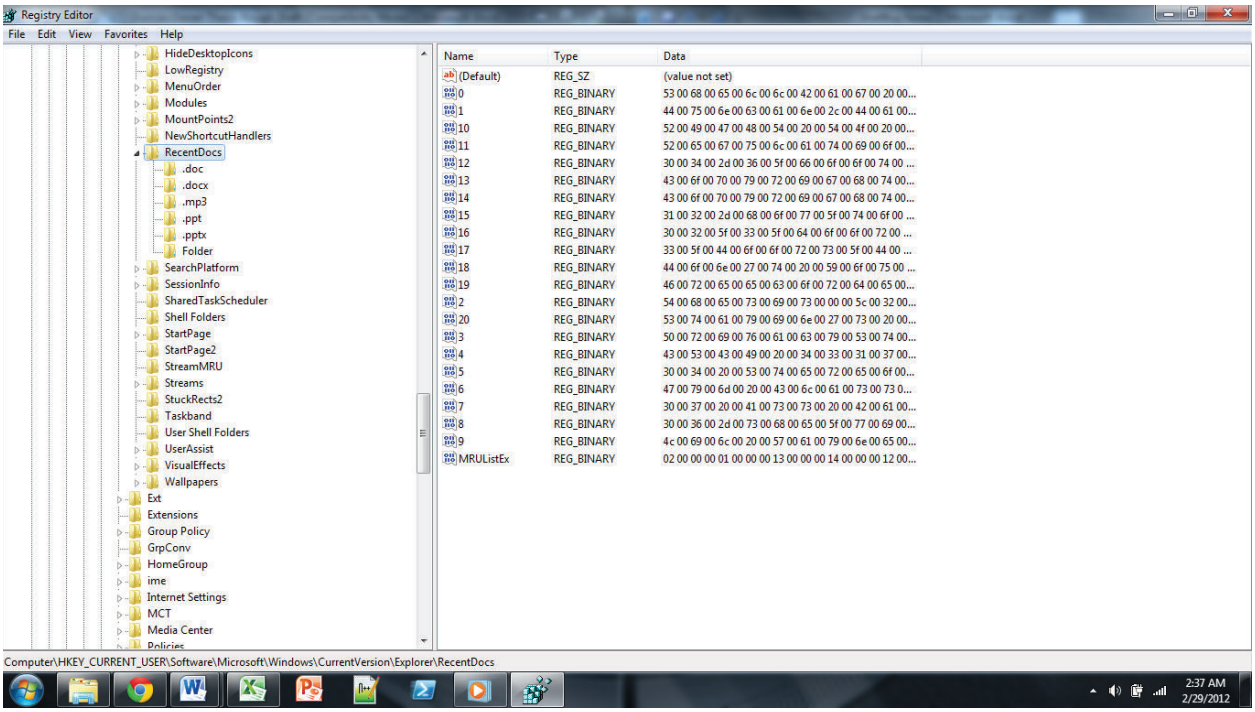


Figure 4: The contents of the *RecentDocs* key are listed above. This information will prove very valuable in writing a parsing tool.

The final partition of the ShellBag area is the *StreamMRU* key. The values of this key contain many different Bag structures that have been pieced together. A *StreamMRU* key

would contain, for example, each Bag file that corresponds with the target folder or its ancestor folders to create a normal directory listing or file path (Khatri, 2004).

It is also important to note that virtual machines are not tracked by ShellBag data. The Control Panel, for instance, does not have a place in ShellBag data because it has a permanent path, and thus, cannot be changed (Metz, 2011).

It is convenient that information regarding the structure of ShellBags is already known. However, it is not readily available in a readable format. The information stored in the ShellBag data is very cryptic both because of the nature of updates performed in ShellBag data and because the data is stored as a binary large object, also known as a binary blob. Therefore, many different pieces of information may be stored in a single blob, such as file and folder names. A parser must be utilized to translate this data into ASCII encoded entries, meaning that anyone could read the information in the entries (Davies, 2006). The following section will discuss an application that will read ShellBag data from the *RecentDocs* key and produce a timeline of events that chronicle a user accessing the file system.

APPLICATION

Overview

Due to the fact that there have been numerous applications written, both commercial and open source, that analyze ShellBags from the *Bags* and *BagMRU* keys, this program pulls directly from the *RecentDocs* key to provide a timeline of user events. As a result of research stating the information that ShellBags are capable of holding, the resulting timeline will display a list of the most recently accessed files and folders on a machine, from most recent to the least recent item held in the Bag file's memory. This program is a command line application running from a Windows PowerShell script.

Algorithm

The program starts by getting the properties of the *RecentDocs* key in the path *HKCU**Software**Microsoft**Windows**CurrentVersion**Explorer.* Next, the retrieved values are tested to see if the system has been keeping track of recently used documents. If this returns false, the script ends. If the test returns true, then the script reads in the *MRUListEx* property, which contains pointers to the *RecentDocs* properties. These pointer values are then read in, converted to decimal, and compared until both the current and following bytes equal zero, thus representing a null. This null value ends the current iteration of the loop, moving on to another value. The real values are read in as hexadecimal values. When all values are read in, these values are converted to ASCII and printed out into the resulting timeline.

Source Code

```
#
# ------------------------------------------------------------------------
# File name:                                    recentDocuments.ps1
# Project name:                                 ShellBag Parser
# ------------------------------------------------------------------------
# Creator's name and email:   Daniel Duncan  duncanda@goldmail.etsu.edu
# Creator's name and email:   Billy Overton  overtonb@goldmail.etsu.edu
# Course-Section:                               CSCI 4018-088
# Creation Date:                                2/14/2012
# Date of Last Modification:                    3/30/2012
# ------------------------------------------------------------------------
# Purpose:  To show the content of ShellBags
#
# Input:    none
# Output:   Timeline of recently accessed documents and files
#
# Note:
# 1. The RecentDocs registry key must contain entries for the script to return
#    the timeline
#/

#Read in the registry value
$a = Get-ItemProperty -path
"HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs"


# See if the registry locaton exists (if it doesn't, then they are not keeping track of
# recently used documents
if ($a) {

   # Grab the MRUListEx key values. They are in reverse byte order and are 2 bytes long
   $mrulistex = $a.psobject.Properties["MRUListEx"].Value

   # This will hold the hex values that point to the differnt reccently used documents
   $mrulistexArray = @()

   # Create an array of hex values that correspond to the keys in the hive
   $i = 0
   while ($i -lt $mrulistex.length -and $mrulistex.length -ge 4) {
      $tempstring = "{0:x2}" -f $mrulistex[$i+3]
      $tempstring += "{0:x2}" -f $mrulistex[$i+2]
      $tempstring += "{0:x2}" -f $mrulistex[$i+1]
      $tempstring += "{0:x2}" -f $mrulistex[$i]
```

```
    $mrulistexArray += $tempstring
    $i += 4
  }

  # Write a header
  WRITE-HOST "`nRecently Used Documents"
  WRITE-HOST "----------------------"

  # Loop through each key listed by the mrulistex list
  $i = 0
  while($mrulistexArray[$i].CompareTo("ffffffff")) {

    # Read in the value, Powershell reads them in as decimals
    $decimal = $a.psobject.Properties[([CONVERT]::toint32($mrulistexArray[$i],16))].Value

    $hexarray = @()

    # I only wanted to grab the filename. So I just read a byte at a time till I find a null.
    $j = 0
    for($j -lt $decimal.length) {

      #break at the first null character
      if($decimal[$j+1] -eq 0 -and $decimal[$j] -eq 0) { break }

      # Grab a byte
      $tempstring = "{0:x2}" -f $decimal[$j+1]
      $tempstring += "{0:x2}" -f $decimal[$j]

      # And add it to the hex array
      $hexarray += $tempstring

      # Jump forward a byte
      $j+=2

    }

    # Remove these external WRITE-HOST statements for decent looking output.
    WRITE-HOST "`t- " -nonewline
    $namearray += $hexarray | foreach {WRITE-HOST –object (
[CHAR][BYTE]([CONVERT]::toint32($_,16))) –nonewline }
    WRITE-HOST ""

    $i++
  }

}
else {
  WRITE-HOST "There are no recently used documents in the registry"
}
```

CONCLUSIONS

Program Results

The aforementioned script proves that the *RecentDocs* registry key can produce a timeline of user events, much like the keys in the two other ShellBag locations. The script does not provide file/folder creation, modification, and access times, but the items given to the user provide enough evidence to convict a criminal.

Summary

In light of research done in the field of digital forensics, it is possible for a forensic examiner to analyze the Windows Explorer settings of a person's computer to determine what actions took place in the file system of that computer. There are several relevant truths that can be gleaned from this research. First, there have been some modifications to the registry in Windows 7 that make it different from XP or any of its predecessors, such as including data for remote and local folders in the same *Shell* key. Second, ShellBag data is only created after the new folder has been closed, not when it is first opened. Third, and most crucial to the forensic process, this data is never removed from the registry, even when the corresponding folder has been removed from the file system. Fourth, and almost equally as important, this ShellBag data contains the short and long folder names, and times for creation, modification, access, and deletion. Fifth, every folder in the file system begins with the Desktop as the root. Finally, the

full path for the target directory can be gleaned from ShellBag data to show the location of the folder in the file system.

All of these facts come together to prove that ShellBag data can be instrumental in proving the guilt or innocence of a suspect, as it provides a collective listing of all folders that have ever been present on a computer. By proving that a specific folder exists, the suspect can be judged on that evidence alone. Although this method of data recovery is susceptible to countermeasures such as “registry cleanup” applications that delete unused or erroneous registry keys, ShellBag data has proven to be a clever auditing mechanism built into Windows operating systems.

WORKS CITED

Davies, P. (2006). Forensic analysis of the windows registry computer forensics. In *Docstoc* docstoc.com. Retrieved from http://www.docstoc.com/docs/2381418/Forensic-Analysis-of-the-Windows-Registry-Computer-Forensics

Hay, A. S. (2004, December). *WRA guidance*. Retrieved from http://mysite.verizon.net/hartsec/files/WRA_Guidance.pdf

Khatri, Y. (2011, October 04). [Web log message]. Retrieved from https://42llc.net/?p=385

Metz, J. (2011). *Windows shell item format*. Unpublished manuscript, Computer Science, , Available from download.polytechnic.edu.na. Retrieved from http://download.polytechnic.edu.na/pub4/download.sourceforge.net/pub/sourceforge/l/project/li/liblnk/Documentation/Windows Shell Item format/Windows Shell Item format.pdf

Microsoft Support. (2008, February 04). *Windows registry information for advanced users*. Retrieved from http://support.microsoft.com/kb/256986

US-CERT. US-CERT, (2008). *Computer forensics*. Retrieved from US-CERT website: http://www.us-cert.gov/reading_room/forensics.pdf

Wikipedia contributors. (2011). Binary large object. In *Wikipedia, The Free Encyclopedia* Retrieved from http://en.wikipedia.org/wiki/Binary_large_object

Zhu, Y., Gladyshev, P., & James, J. (2009). Using shellbag information to reconstruct user activities. *Digital Investigation*, (6), 69-77. Retrieved from http://www.dfrws.org/2009/proceedings/p69-zhu.pdf