

CASE STUDIES: USER HIVES

INFORMATION IN THIS CHAPTER

- NTUSER.DAT
- USRCLASS.DAT

Introduction

When first I sat down to write this book, it occurred to me that this chapter...one about tracking user activity...might be the most useful and interesting chapter. Windows does a great job of providing a quality experience to the user, keeping track of documents they had opened, saved, or accessed, how they had set up and configured their favorite Solitaire game, which web browser they used, which application is launched when the user double-clicks on a file in the shell, and even the size and position of various application windows on the desktop. All of this information has to be tracked somehow, and for the most part, a great deal of it is tracked through the user's Registry hive files. The fact that this information is recorded in any manner at all is transparent to the user, but for a knowledgeable analyst, the Registry, and in particular the user's hives, can be veritable treasure trove of forensic data.

In the previous chapter, we discussed several of the Registry hives that pertain most directly to the system; the SAM, Security, System and Software hives. In this chapter, we will be focusing primarily on two hives found within the user profile directory; the NTUSER.DAT hive and the lesser known USRCLASS.DAT hive. These two files, to varying degrees based on the version of Windows being examined, can provide a great deal of data regarding the user's activities on a system. In this chapter, we're going to take a look at the various ways this information can be used, and more importantly, how it can be used effectively to support a number of types of investigations.

As with the previous chapter, this chapter should not be considered a comprehensive and complete list of all possible Registry keys and values that might be considered important or valuable to an analyst. While Windows XP systems are fairly well understood, there is still a lot about Vista systems, and now Windows 7 systems, and even Windows 8 and 10 systems, that require a great deal of

research, particularly in the area of Registry analysis. Add to that the proliferation of applications on these systems, and there's an apparent never-ending supply of Registry locations that can be of value, including (but not limited to) used by malware to maintain persistence on the system. Rather than providing a long list of Registry keys and values of interest, it's more important to understand how some keys and values can be used, not only by an intruder or malware author but more so by a forensic analyst in order to paint a more complete picture of an examination. Understanding how the user hives can be used is far more important than maintaining a long list of keys and values that don't have any context or anything to indicate how they're important.

A final thought before we head into this chapter; as with previous chapters, the most important aspect of Registry analysis is to first understand your goals; what are you looking for or trying to prove. Many analysts kick off an examination by loading Registry hives into a viewer, without really understanding what it is they're looking for; this will often result in "no findings" and a great deal of time spent finding this out. If you understand what you're interested in and what you're looking for, you can find it very quickly.

NTUSER.DAT

The main Registry hive for the user is the NTUSER.DAT hive, located in the root of the user's profile folder.

One of the things we'll be addressing in this chapter is referred to as a "most recently used" list, or "MRU." This sort of list usually appears as values beneath a Registry key (and can appear in other artifacts and data sources, such as Jump Lists), and most (if not all) of them are found within the user hives. The idea behind an MRU, from the perspective of a forensic analyst, is that there is some mechanism for differentiating which of the events or actions occurred most recently. An example of an MRU may be several values that point to image files the user accessed via a specific application. The most recently accessed (or "used") file may be identified by having a specific value name or by reviewing a separate value named "MRUList" or "MRUListEx". What having an MRUList means is that there is some sort of ordered numbering scheme that is used to track the entries, and in the case of some values, there may also be another value named MRUList or MRUListEx that will tell you the order of the MRU values. In some cases the values are given numbers as names ("0000", "0001", etc.) as they are added to the key; the most recent value is named "0000" and when the next value is added, it is named "0000" and the previous value is "pushed down" to "0001", and so on. This way,

looking at the value names, you can get a very quick view of the order in which the values were added, and there is no need for an MRUList (or MRUListEx) value.

In other instances, the values are assigned numbers as names (which may begin with the letters “MRU”, depending upon the key and the application that uses them), and there will be an additional value named “MRUList” or “MRUListEx” that maintains the order in which the values were “used” (again, this depends upon the application). For example, consider a Registry key for which the first value added is named simply “a”. At this point, the MRUList value would indicate that the “a” value was the MRU value. At some point, several other values (b, c, etc.) are added, and the MRUList value indicates the order accordingly (c, b, a). However, at some point the user does something that reuses the first value (conducts a search for the same keyword, accesses the file, etc.); the MRUList value would now indicate that the MRU order is now “a”, “c”, “b”. Even though all of the values keep their original names, the MRUList value indicates the order in which the values were “used.”

Throughout this chapter, we’ll be looking at a number of MRU lists, and in each case, we’ll discuss the means for identifying the most recently accessed file or used item. The point is that while an MRU list may maintain a list of recently accessed objects, the means for identifying the most recently accessed object may vary and be different.

System Configuration Information

While most of the system configuration information found in the Registry is maintained in other hive files (as discussed in chapter: [Analyzing the System Hives](#)), the user’s NTUSER.DAT hive file does contain some configuration information, as well, particularly as it applies to applications.

REDIRECTION

On 64-bit versions of Windows, 32-bit applications are redirected to another within the Registry (there’s a similar function for the file system, but that is a topic that’s beyond the scope of this book). Suffice to say that there are number of keys discussed in this chapter, as well as the previous one, that have corresponding values beneath the *Software\Wow6432Node* subkey, such as the Run key discussed later in this chapter. Registry redirection is discussed at length at the Microsoft website and can be found online at [https://msdn.microsoft.com/en-us/library/windows/desktop/aa384232\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa384232(v=vs.85).aspx).

AutoStart

The autostart category refers to those locations within the user's hives that permit applications to start automatically (yes, I know that definition is somewhat circular), with no interaction from the user beyond logging in, or launching an application.

The Run Key

I usually refer to this key (the key path is \Software\Microsoft\Windows\CurrentVersion\Run) as “the ubiquitous Run key,” because it seems to be used so often by bad actors to ensure the persistence of some form of malware. Often times, this is predicated upon the privileges of the user account being infected; if the user account has administrator privileges, the malware may persist via a Windows service, but if the user account has normal user privileges, the malware may persist via the user's Run key.

Values beneath the Run key are all run asynchronously without any interaction from the user beyond logging into the system. Unless the application being run has some method for doing so, the user sees no notification that any particular program is running. Further, if the file pointed to by the value data doesn't exist (say, it was deleted), the user receives no notification, either.

With respect to the user's hive in particular, the functionality of this key is very often misrepresented in a variety of online sources, which tends to renew the cycle of this key being misunderstood. When an application persists via this key, by having a value beneath the key whose data points to the application executable, that application will only start following a system reboot *when the user logs in*. In a way, it's understandable how this sort of mistake is made, as Microsoft makes that statement themselves in their malware write-ups; for example, consider the write-up regarding Win32/Cypaux (found online at <https://www.microsoft.com/security/portal/threat/encyclopedia/Entry.aspx?Name=Win32%2fCypaux#tab=2>). In that write-up, the author states that persistence is achieved via the user's Run key and states that it's used to “ensure that its executable runs at each Windows start.”

CORRELATING NEW VALUES

Many times when conducting forensic analysis of a system, I'll look at the contents of the Run key for a particular user and see several values beneath the key, sometimes as many as six or seven. The LastWrite time of the key tells me when the contents of the key were last modified, but there isn't anything that's readily available within the key structure that will tell me what specifically was modified.

CORRELATING NEW VALUES—Cont'd

One means for getting some context with respect to which value may have been added would be to create a timeline of system activity, including file system and Windows Event Log metadata, as well as Registry key LastWrite times. This may illustrate a cluster or series of artifacts that illustrate malware being installed on the system, malware that uses the user's Run key for persistence.

Another means for determining which value was added, or removed, would be to compare the contents of the key to what you find in the NTUSER.DAT hive for the user in a Volume Shadow Copy.

A good deal of malware can be seen using the user's "Run" key for persistence, particularly when the user account that gets infected has only normal user (not administrator) privileges. At the 2012 SANS Forensic Summit, Elizabeth Schweinsberg gave a presentation on Registry analysis (her presentation can be found online at <http://digital-forensics.sans.org/summit-archives/2012/taking-registry-analysis-to-the-next-level.pdf>). During her presentation, she shared some fascinating statistics on the use of the Run key by malware, which she collected by "trawling" the Symantec website for Registry keys. From what she was able to collect at the time, the "Run" key was used 42.2% of the time when malware persisted within the user's NTUSER.DAT. This is really very interesting, but at the same time, we have to keep in mind that it's not abundantly clear how the original data is collected, nor how the malware is executed in order to collect that data. For example, if the malware is executed with normal user privileges, it may be more likely that the malware will persist via the user's "Run" key.

As such, when examining the values beneath this key, you'd want to look for things such as odd file paths, with files in the root of the user profile (ie, C:\Users\harlan\), the user's "temp" folder (ie, C:\Users\harlan\AppData\Local\Temp), etc. Not only can the *user_run.pl* RegRipper plugin be used to extract the contents of the user's Run key, but other plugins can also be used to look for specific values. For example, the *ahaha.pl* plugin looks for indications of the malware that persists using the value name "360v", and the *reveton.pl* plugin looks for indications of the Reveton malware (a description of this malware can be found online at <http://www.microsoft.com/security/portal/threat/encyclopedia/Entry.aspx?Name=Win32%2fReveton#tab=2>) persisting via the user's Run key.

WHAT TO LOOK FOR

Sometimes folks will ask me what I tend to look for when looking at values beneath the Run key. Back in the early 2000s, my response was largely based on experience; there were value names or data values I could look

Continued

WHAT TO LOOK FOR—Cont'd

at that just didn't "feel right." Over time, as I worked a number of similar cases, what I saw as unusual would be something that I had seen before and knew to be "bad." Even now, at a first glance, I look for odd or random value names, data paths such as "C:\ProgramData", "C:\Users\user\AppData\Local", "C:\Users\user\AppData\Local\Temp", etc.

The RunOnce Key

The RunOnce key can be found in the path *Software\Microsoft\Windows\CurrentVersion\RunOnce*. Values beneath this key will be launched the next time the user logs in, and then the value will be deleted once the command has completed. However, according to Microsoft's Knowledge Base (KB) article 137367 (found online at <https://support.microsoft.com/en-us/kb/137367>), if the value name is prepended with an "!", the value will not be deleted once the command has completed. If this extra step is taken, the RunOnce key takes on the capabilities of the Run key described above.

The *user_run.pl* RegRipper plugin will collect values from both the Run and RunOnce keys, as well as the corresponding keys beneath the *Wow6432Node* key.

TEMPORAL PROXIMITY

The term "temporal proximity" is a Star Trek-y kind of term I first heard used in the fall of 2008 by Aaron Walters (of Volatility fame), and it refers to when response activities start in relation to the incident having occurred. I bring this up because keys such as the RunOnce key can really illustrate the importance of temporal proximity, as well as rapid incident detection and response. Something that differentiates the RunOnce key from the Run key is that items listed in the RunOnce key are run once. Any command line listed as a value beneath this key will be run the next time a user logs into the system and be deleted (before or after being run, per our previous explanation). The value of temporal proximity is also illustrated by issues such as deletions; when a Registry key is deleted, the space used by the key becomes part of the unallocated space of the hive file and may be reused (ie, overwritten) at some point (the same concept that applies to files in the file system). The sooner response activities are initiated, the more likely you are to have access "fresh" data.

Other AutoStart Locations

There are several other autostart locations similar to the Run key, which are listed as keys to check within the *user_run.pl* RegRipper plugin. Some of these locations can be referred to as "legacy" Run keys, but needless to say, they are still effective because they work. For example, one such key is the *Software\Microsoft*

Windows\CurrentVersion\Policies\Explorer\Run key, and it's corresponding key beneath the *Wow6432Node* subkey. Also, beneath the *Software\Microsoft\Windows NT\CurrentVersion\Windows* key, the “load” and “run” values also provide persistence locations that can be, and have been, used by malware authors.

The consulting company Cylance also has an interesting blog article available regarding persistence locations, which can be found online at <http://blog.cylance.com/windows-registry-persistence-part-2-the-run-keys-and-search-order>.

Program Execution

As was discussed in chapter “[Analyzing the System Hives](#),” those values within the Registry that serve as autostart locations also serve to provide information regarding program execution as well, as long as the condition for that location has been met. For example, in the previous section we discussed the *Run* key as an autostart location; as such, the user logging in would be the condition that needed to be met, and as such, would serve to indicate that the program had been executed.

Applets

Windows systems ship with a number of small applications (referred to as “applets”) installed and readily available to users. These applets include RegEdit, MS Paint, the System Tray, and MS Write, and the key path is *Software\Microsoft\Windows\CurrentVersion\Applets*. [Fig. 4.1](#) illustrates the contents of a user's Applets key.

As you can see in [Fig. 4.1](#), the Paint and Wordpad applets include a subkey called “Recent File List”, each of which provides an MRU listing of files that the user accessed via the applet. [Fig. 4.2](#) illustrates what the contents of the “Recent File List” key beneath the *Applets\Paint* key might look like for a user.

As you can see in [Fig. 4.2](#), the MRU values are listed, as we discussed with MRU values in general, there are no time stamps with each of the individual MRU entries. While you can see that the user accessed a file using a particular application (in this case, MSPaint), this data by itself will not tell you *when* the user accessed the file. To determine when the user accessed the file, you would need to look to additional information, perhaps even resources outside of the Registry (such as Jump Lists).

The RegEdit applet key does not include an MRU list, but it does include a *LastKey* value, as illustrated in [Fig. 4.3](#).

These keys beneath the “Applets” key illustrate the user's use of the programs and therefore provide indications of program

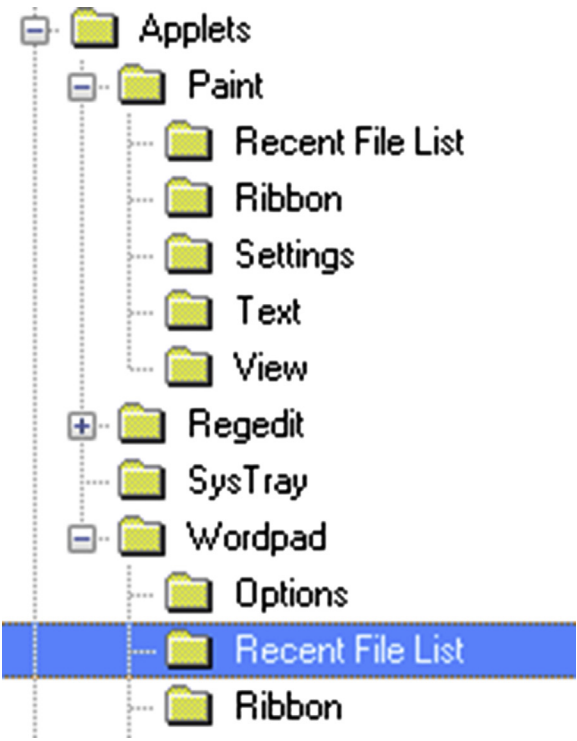


Figure 4.1 User's Applets key, showing subkeys.










Value	Type	Data
 File1	REG_SZ	C:\Users\harlan\Desktop\herrcore.jpg
 File2	REG_SZ	C:\Users\harlan\Desktop\usb.jpg
 File3	REG_SZ	C:\Users\harlan\Desktop\tab.png
 File4	REG_SZ	C:\Users\harlan\Desktop\ltn.png
 File5	REG_SZ	D:\books\WRF\ch3\2e\fig3.sticky.tif
 File6	REG_SZ	C:\Users\harlan\Desktop\weather.jpg
 File7	REG_SZ	C:\Users\harlan\Desktop\photo.PNG
 File8	REG_SZ	D:\books\WRF\ch3\2e\fig3.3.TIF
 File9	REG_SZ	D:\books\WRF\ch3\2e\fig3.amcache5.tif

Figure 4.2 MRU values beneath user's "Paint\Recent File List" key.

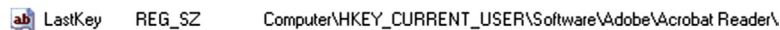


Figure 4.3 Partial contents of user's LastKey value.

execution. As is the case with a number of artifacts, these keys, particularly those for the MSPaint and Wordpad applets, can also provide indications of user access to files. The LastKey value, however, can provide some very useful indications of user activity, particularly when individual indicators within a cluster are apparently absent.

THE LASTKEY VALUE AND ADMIN CLEANUP

I was performing forensic analysis of several systems during a targeted threat response engagement and ran across some anomalous findings on two particular systems. We had some indications that two specific systems had been infected with the malware in question, but what we weren't seeing was the persistence mechanisms; in this case, Registry keys associated with a Windows service. I started by looking for alternate means of persistence but then had a hunch and checked the administrator's LastKey value. In both cases, value contained the path to the service that followed the malware persistence path alphabetically. Doing some testing, I was able to easily verify that the administrator had opened the Registry Editor, navigated to the Windows services key used by the malware for persistence, and deleted it. Prior to closing the Registry Editor, the next key in alphabetical order was "in focus", and that was the key that was recorded in the LastKey value. Other artifacts on the system validated the finding that the administrator had taken steps to try to "clean up" the malware infection on those two systems.

SysInternals

Microsoft's SysInternals tools (found online at <https://technet.microsoft.com/en-us/sysinternals/bb545021.aspx>) include a number of very useful tools for administrators, and like many other utilities that are useful to administrators, they can also be useful to those with nefarious purposes. However, the tools include a small bit of functionality that can make them something of hindrance when employed but is extremely valuable for forensic analysts. In order to use the tools for the first time on a system, the user has to accept the end user license agreement (EULA); this is easily done by including the "/accepteula" argument at the command prompt or by clicking on the appropriate button that appears in the dialog box that appears if the "/accepteula" switch is not used. Not only do you have to employ this specific switch but also you have to spell it properly. When the EULA is accepted, a Registry key is created for the application in the user's hive, as illustrated in Fig. 4.4.

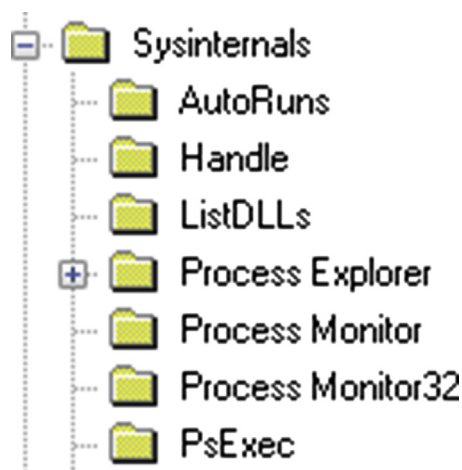


Figure 4.4 SysInternals tools listed in the NTUSER.DAT file.

If the `strings.exe` utility is used (I tend to use this particular utility quite a bit) and the EULA is accepted, then a key is created in the following path:

```
Software\SysInternals\Strings
```

The key usually has just one value (“EulaAccepted”), but the key’s LastWrite time correlates to when the utility was run first run. How is this useful? Consider an investigation (violation of acceptable use policy, abuse of admin privileges, data breach) where the `PSEXec` tool was used. This tool is a remote execution utility and allows someone with the appropriate privileges to run commands on remote systems. In a corporate environment, this might be a domain admin, or someone who has access to admin-level credentials on the remote systems. The `PSEXec` tool is a command line tool and does not have a graphical user interface (GUI); regardless, it is a tool that is designed to run on Windows systems and has been used legitimately by system administrators to perform various functions. Unfortunately, it has also been used by dedicated adversaries to move laterally within a compromised network. When it’s run for the first time from the source system, a Registry key (path is “Software\SysInternals\PSEXec”) is created, leaving an indicator of the use of the tool.

COMMAND LINE TOOLS

I’ve been performing incident response and digital forensic analysis since about 1999 or so, and I’ve seen time and again that intruders are very familiar with the use of command line tools, particularly because the level of access required to use command line tools is relatively easy to achieve, and the use of such tools usually allows them to pass under

COMMAND LINE TOOLS—Cont'd

the radar of most system administrators. This is the reason why I highly recommend to...well, anyone who will listen...that they install a process creation monitoring tool such as Microsoft's own Sysmon or Bit9's Carbon Black. Either of these tools, when properly employed, provide a level of monitoring that is similar to having a video of a real-world crime occurring. Don't get me wrong...such tools are not the "silver bullet" for security, but in my experience, having such tools in place can enhance detection of an intrusion to significant levels and reduce the overall time for response from days and weeks to just minutes.

UserAssist

During a job interview a number of years ago, the interviewer asked me what my favorite Registry key was; if I had to answer that question today, I'd have to say that it is the UserAssist key. Oddly enough, the key name is pretty descriptive...the contents of this key assist the user. Okay, I know it's a stretch but bear with me; beneath this key (we'll address exactly where shortly) are Registry values that track a user's interactions via the Windows Explorer shell, primarily when the user clicks or double-clicks on certain items. This information is then used by the operating system to tailor the user experience; for example, like many other folks working in corporate America, I used to use a Windows XP SP3 laptop for work. Each morning when I would log in to the corporate network, I would click on the Start button, go to Programs, then to "Microsoft Office", and in the final menu, I click on Microsoft Outlook. After the first couple of times that I did this, when I got to that final menu, only the Microsoft Outlook choice was immediately visible; why would the operating system completely expand all of the menus in the path, when I'd demonstrated that I was primarily interested in only one or two items? It's a much better and preferable user experience to show those items I'm most interested in via customized menus based on my usage history. Given this, and the data included in the relevant values, would lead you to believe that this key should really be called "Forensics Assist"!

So, to begin, the UserAssist key (within the NTUSER.DAT hive) is located in the following path:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\
UserAssist
```

Beneath this key (on all versions of Windows), you'll find two (I've seen three on a very few Windows systems; as I'll describe shortly, there may be more) keys with names that appear to be globally unique identifiers, or "GUIDs," as illustrated in [Figs. 4.5 and 4.6](#).

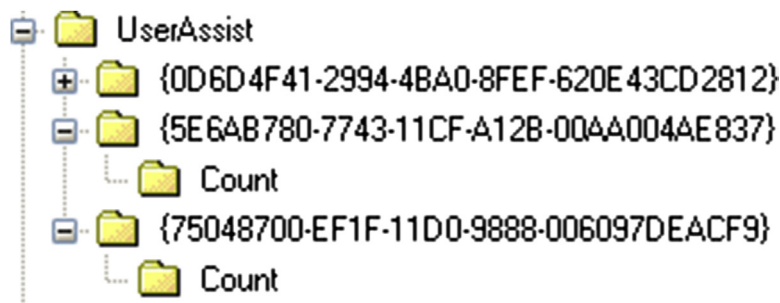


Figure 4.5 Windows XP UserAssist key.

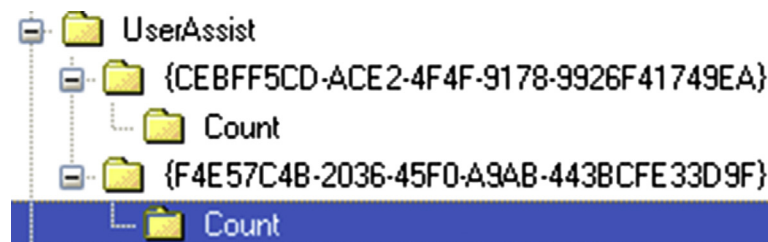


Figure 4.6 Windows 7 UserAssist key.

OTHER VERSIONS OF WINDOWS

I do not have access to an extensive library of Windows systems, but within the NTUSER.DAT file from one Windows 10 system, I saw nine subkeys (each with a name that looked like a GUID) beneath the UserAssist key; within the NTUSER.DAT hive from a Windows Server 2012 system, I counted a dozen subkeys with GUIDs for names. Clearly, there's a great deal of information being maintained in these keys, as well as a great deal of additional research that needs to be done.

As you can see in Figs. 4.5 and 4.6, each of the keys with GUIDs for names will have a subkey named “Count”, and we're interested in the values located within the Count subkeys. Fig. 4.7 illustrates what these values look like in a Registry viewer.

The values illustrated in Fig. 4.7 don't look very useful, do they? Well, that's because the value names are encoded via the ROT-13 substitution algorithm; that is, each letter is swapped with the one 13 positions further down in the alphabet. To undo (decrypt) the algorithm, we simply reverse the substitution. Fortunately, the *userassist.pl* RegRipper plugin will handle this translation easily using the following code:

```
$value_name =~ tr/N-ZA-Mn-za-m/A-Za-z/;
```

[illegible]

Figure 4.7 UserAssist\...\Count key values.

Before we proceed, it's important at this point to mention that Didier Stevens has invested a considerable amount of time and effort in researching the values beneath the UserAssist key, particularly with respect to Windows 7. Without question, Didier deserves a great deal of credit for the current understanding of, and interest in, the contents of the UserAssist key within the computer forensics community. Didier's findings and tool can be found online at <http://blog.didierstevens.com/programs/userassist/>.

VIGENERE ENCRYPTION

During his research into the UserAssist key, Didier discovered that in the beta version of Windows 7, rather than ROT-13 “encryption,” the value names were encrypted using Vigenere encryption, a polyalphabetic substitution cipher originally described by Giovan Battista Bellaso in 1553. The final release of Windows 7 switched back to the use of ROT-13 encryption. If nothing else, this illustrates how understanding the version of Windows that you’re examining is absolutely critical.

According to a Microsoft employee I spoke with, the use of the encryption or obfuscation technique isn't to protect any sensitive information; rather, it's intended as a deterrent to prevent the user from modifying any information in the value name or data. From what I've seen over the years, this obfuscation technique does little more than obviate the use of text searches via a commercial forensic analysis framework and requires the analyst to parse the data first before running their search.

Okay, so how is all of this important? Well, remember that the operating system uses some method for keeping track of a user's actions (which items they click on, which shortcuts and applications they access, etc.) and then uses that information to provide an improved (beyond the default installation) experience to the user. Both testing and analysis indicate that the information embedded within the binary data associated with many of the

values beneath the UserAssist key includes a 64-bit time stamp (ie, our familiar FILETIME structure), as well as a counter (referred to as a “run count”) that appears to indicate how many times the user has interacted with the shell in the manner in which these values would be created or modified.

RUN COUNT

When the counter value embedded within the UserAssist value binary data was first examined, it appeared that the count actually started at 5, rather than 0. There seemed to be no apparent reason for this (the internals of any algorithms that may use this information are not known), Ovie Carroll and Bret Padres (of the CyberSpeak podcast fame; interestingly enough, after a long absence, Ovie recorded another podcast on August 31, 2015) came up with a very funny mnemonic device; the name “Gates” (as in “Bill Gates”) contains five letters. Regardless of the reason apparently starting the count at 5, testing indicated that this was, in fact, the case; performing an action and then parsing the information on live system would result in a count value of 6 (the first time that the action was recorded, plus 5).

In short, the binary data can be parsed (by RegRipper plugins) to determine how many times the user had taken this action (ie, navigated through the Programs menu to launch MS Word, double-clicked a desktop icon, etc.) via the shell, and when they last did so. An important aspect of this is that in order to create/modify these values, the user needs to interact with the Explorer shell; that is, if the user clicks Start and then types “cmd” into the Run box on Windows XP, you don’t get the same artifacts as if the user clicks Start, Programs, Accessories, and chooses “Command Prompt”, and you won’t be able to “see” what the user did in the command prompt.

Let’s take a look at example; below is an excerpt from the output of the RegRipper *userassist.pl* plugin run against an NTUSER.DAT hive extracted from a Windows XP system:

```
(75048700-EF1F-11D0-9888-006097DEACF9)
Thu Feb 7 13:37:26 2008 Z
  UEME_RUNPATH:E:\FTK Imager.exe (1)
Thu Feb 7 12:41:42 2008 Z
  UEME_RUNPATH:C:\Program Files\Microsoft Office\OFFICE11\
WINWORD.EXE (120)
Thu Feb 7 11:27:41 2008 Z
  UEME_RUNPATH:C:\WINDOWS\regedit.exe (5)
Thu Feb 7 10:39:55 2008 Z
  UEME_RUNPATH:Lotus Notes 7.lnk (142)
  UEME_RUNPATH:C:\Program Files\Lotus\notes\notes.exe (142)
Thu Feb 7 10:38:38 2008 Z
```

```

UEME_RUNPATH:C:\Program Files\AT&T Network Client\
NetClient.exe (147)
UEME_RUNPATH:{5D5A8163-501D-4F38-8B17-23488A324D64} (146)
UEME_RUNPATH:{AC76BA86-1033-0000-BA7E-100000000002} (112)

```

As you can see from the above excerpt, the *userassist.pl* plugin decrypts the value names beneath the UserAssist subkeys and then, where applicable, parses the associated binary data for the run count, and the last time the action was taken. First, we see the GUID that we mentioned which is one of the UserAssist subkeys; opening the Software hive from the system from which the NTUSER.DAT hive was extracted in Registry viewer (see chapter: [Processes and Tools](#)) and searching for that GUID, we find that it refers to a class identifier (CLSID) beneath the Classes key that points to “Active Desktop”.

Next, we see an indication that on February 7, 2008, at approximately 13:37:26 Z (see the “Time References” sidebar), FTK Imager was launched from the E:\ drive. Well, that’s where I placed a CD in the system and ran FTK Imager in order to collect specific files from the system, including the Registry hives. That reference begins with “UEME_RUNPATH”, which indicates an executable file was accessed; in this case, by double-clicking the program icon as it appeared in Windows Explorer (opened to the CD, of course). According to the run count (ie, the number in parentheses after the application path), at this point, FTK Imager was only run once.

TIME REFERENCES

Most of the RegRipper plugins report time with “Z” or “UTC” at the end. The “Z” refers to Zulu, or Greenwich Mean Time (GMT). This is analogous to Universal Coordinated Time, or UTC. When performing analysis across multiple systems, or across multiple time zones, normalizing the time stamps to a common format and reference point can make that analysis much easier. I’ve had several cases where an intruder accessed systems within an organizations infrastructure that were dispersed across multiple time zones and normalizing all time stamps on all of the affected systems to UTC made it much easier to follow his trail, and more importantly, illustrate it to the customer.

Next, we see that regedit.exe was launched, and that Lotus Notes (our e-mail application at the time) was run for the 142nd time by double-clicking the Windows shortcut (on the desktop). Beneath that, at 10:38:38 Z, we see that the AT&T Network Client (VPN solution) was accessed, and that there are two GUIDs as well. Once again, opening the Software hive from this system in a Registry view application and searching for “{5D5A8163-501D-4F38-8B17-23488A324D64}”, we find that this also appears as a subkey

name beneath the Microsoft\Windows\CurrentVersion\Uninstall key, and that subkey contains a value named “DisplayName” set to “AT&T Network Client”. The other GUID (“AC76BA86-1033-0000-BA7E-100000000002”) appears in 24 locations (keys and values) throughout the Software hive and appears to refer to the Adobe Acrobat Reader version 7.0 installer.

Other entries may appear with different prefixes in the output of the *userassist.pl* plugin (and other tools). For example, rather than being preceded by “UEME_RUNPATH”, some decoded values may begin with “UEME_RUNPIDL” (a “PIDL” is a pointer to an ItemIdList structure, which is used to identify objects in the Shell namespace [7]), referring to a folder or shortcut, and others may begin with “UEME_RUNCPL”, which refers to Control Panel applets being clicked.

Personally, I’ve used the information within the UserAssist keys to great effect during a number of examinations. I’ve seen where users and intruders have installed and then run the password cracking tool named “Cain.exe”, in order to collect passwords from a variety of applications; even after deleting the application, the entries in the UserAssist key persist. I’ve seen where programs were run from an external resource, such as a CD or thumb drive, because the user double-clicked the icon via the Windows Explorer shell. I’ve also seen where system administrators who stated that once a system had been confiscated and “secured,” they “didn’t do anything” had actually installed, run and then uninstalled two consecutive antivirus (AV) scanning applications, one after another. I guess they were just trying to be thorough... but their actions were “recorded” and accounted from some of the artifacts that I was seeing, as well as some I wasn’t seeing. I’ve seen where intruders have installed malware on systems that we weren’t immediately aware of, and this information helped us a great deal in our examination.

I’ve also examined systems where there were apparent disparities with time stamps recorded on the system, and in parsing the UserAssist key information, found “UEME_RUNCPL” entries referencing “timedate.cpl”, the Date and Time Control Panel applet that allows the user to modify the system time. The user can change the system time in this manner by either double-clicking the Control Panel applet or by right-clicking the clock on the far right of the TaskBar and choosing “Adjust Date/Time” from the context menu that appears.

As we saw in figures 4.5 and 4.6, Windows 7 uses a different set of GUIDs for the UserAssist subkeys, and that’s not all that’s different. Those values that contain time stamp data are also formatted differently and appear to contain a great deal more information. Again, Didier Stevens has some testing and analysis in this area,

in an attempt to identify the various pieces of information (ie, such as how long the application had focus, etc.), and reviewing some of what he’s published, it’s easy to see how an analyst can use them to support his findings during an examination. This is an area that will require significantly more research and testing.

One final note with respect to the UserAssist key; there have been two additional Registry values identified that may significantly affect the information maintained beneath the UserAssist subkeys. Both of these would be values added (they do not exist by default on any system I’ve seen) to a Settings key (beneath the UserAssist key). The first value, NoEncrypt, when set to a DWORD value of “1”, can apparently be used to disable the ROT-13 encryption. The other value, NoLog, when set to a DWORD value of “1”, can apparently be used to disable logging all together. Remember, though...if the logging or recording of user interaction data is disabled, the user experience will be significantly altered, as data used to enable customized menus based on usage history is no longer available. Now, I haven’t seen either of these values during an engagement, but they are important for an analyst to be aware of, as the absence of entries beneath the UserAssist subkeys could be the result of deletion (manually or via an “evidence eraser” program or script) or through the addition of the NoLog value.

NOINSTRUMENTATION

Another Registry value mentioned in MS KB article 292504 (found online at <https://support.microsoft.com/en-us/kb/292504>) is “NoInstrumentation”. This is a value that can be set via Group Policies and would be added to the user’s CurrentVersion\Policies\Explorer key. When set to a DWORD value of “1”, this value will “prevent the system from remembering the programs run, paths followed, and documents used”; apparently, this value may have more wide-ranging effects than simply disabling recording of information beneath the UserAssist key.

Application Compatibility Assistant

Windows systems have something referred to as the “program compatibility assistant” (PCA), which can be used to detect runtime issues in older applications. I won’t go into detail in this section as to the specifics of how PCA works; if you want more information, one resource can be found online at <http://blogs.technet.com/b/askperf/archive/2007/10/05/the-program-compatibility-assistant-part-two.aspx>.

The user’s NTUSER.DAT may contain information about applications that had run on the system and were monitored by PCA, even if PCA did not detect any issues with the application. For example, the key path *Software\Microsoft\Windows NT\Current*

Version\AppCompatFlags\Compatibility Assistant\Persisted is specifically intended to store a list of programs for which PCA came up, but no compatibility modes were selected. The values beneath this key point to applications that had been run on the system and for which PCA came up, but the contents of this key are not an MRU list, and there is no time stamp information associated with each value. As such, you can see that an application was run but not when it was run.

In December 2013, Corey Harrell wrote an excellent blog post that explains what PCA is and what it does (that post can be found online at <http://journeyintoit.blogspot.com/2013/12/revealing-program-compatibility.html>). In that post, Corey also mentioned the *Software\Microsoft\Windows NT\Current Version\AppCompatFlags\Compatibility Assistant\Store* key, which appears to be similar to the Persisted key, but specifically for Windows 8, and meant to replace the use of the Persisted key. I should note that on the system on which I'm writing this book, both the Store and Persisted keys exist; the system was originally a Windows 7 system, but I upgraded (as opposed to doing a clean reinstall) to Windows 10.

Both of these keys (depending upon the version of Windows you're examining) may provide to be extremely valuable in terms of determining applications launched within the user context. As the values beneath these keys are generated by the operating system itself as the application executes within the ecosystem of the OS, they are not something that the application specifically controls. As such, a user running an unauthorized application may result in indicators that persist well beyond the deletion of the application itself and be valuable for forensic analysis.

Terminal Server Client

I've been involved in a number of incident response engagements where we found that a dedicated adversary was able to access an infrastructure through the use of Terminal Services; as such, their access to systems was via a remote desktop. As they were interacting with the Windows Explorer shell, we were able to follow their activities in much the same manner as if they had been sitting at the keyboard.

Essentially what we'd seen after developing a timeline of user activity is that the adversary had accessed some systems and then run the Terminal Server Client (formerly known as the Remote Desktop Protocol, or "RDP") to leap frog and move laterally to other systems within the infrastructure. This was very fortunate for us, as it not only helped us scope the incident but also provided a time frame for the access, allowing us to focus our analysis of the remote system to a specific time window. This can be extremely

valuable when the adversary is using an administrator's account and accessing systems outside of the normal working hours for that administrator.

I wrote, and use, the *tsclient.pl* and *tsclient_tln.pl* plugins to collect information from the *Software\Microsoft\Terminal Server Client* subkeys, the latter doing so in a format useful for including in a timeline of system activity.

REMOTE DESKTOP TOOLS

Just as Hamlet said, "There are more things in heaven and earth...", there are other ways to access the desktop on remote systems, using applications such as WinVNC and its variants. The RegRipper plugins *realvnc.pl*, *vncviewer.pl*, and *winvnc.pl* can help an analyst determine if any of these tools had been used, and if so, which remote systems they were used to access.

Malware

As we've discussed throughout this chapter, as well as previous chapters, some malware will create Registry keys during their installation which they do not use for persistence. For example, the Symantec write-up on the Korplug backdoor malware from 2013 (found online at http://www.symantec.com/security_response/earthlink_writeup.jsp?docid=2015-030203-1048-99) describes the malware creating the "SXLOC.ZAP" value within the user's NTUSER.DAT hive. A Sophos technical paper (found online at <https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/pluginx-goes-to-the-registry-and-india.pdf?la=en>) on a similar variant from the same malware family describes the use of the same value. I wrote the *malware.pl* plugin to check for the keys and values associated with malware that left indications in the Registry hives not used for persistence.

There was some malware a while back that used an interesting means of persistence; I'm including it in this section simply because I couldn't find a more suitable location within the chapter. The malware was dubbed "Win32/KanKan" (a detailed analysis of the malware can be found online at <http://www.welivesecurity.com/2013/10/11/win32kankan-chinese-drama/>) and was found to use a Microsoft Office AddIn to persist on an infected system, ensuring that the malware would be loaded into memory each time the appropriate Office application (MS Word, Excel, etc.) was launched. I wrote the *kankan.pl* plugin specifically to look for and report on these addins.

Keep in mind that Registry values (and in some cases, keys) may be created as the malware interacts with the Windows operating system environment. These values are not created by the malware; there's nothing in the actual code for the malware that includes instructions to create these values or data entries. Rather, they're created as a result of the malware running within the Windows operating system environment.

File Access

There are a number of times when determining a user's access to files, or more specifically, the user account used to access files, can be paramount. This can be pertinent information during a human resources issue, access to illicit images or files case, as well as during data breach cases involving a targeted, dedicated adversary.

One of the great things about Windows systems from the perspective of a forensic analyst is that the systems record and save a great deal of information specific to actions taken via a user account. This can be valuable, as the information is maintained on the system long after the file has ceased to exist on the system.

RecentDocs

Windows systems do a very good job of tracking what documents a user has accessed, making them available in the Recent Documents menu, as illustrated in [Fig. 4.8](#).

This list of documents can be very revealing about a user's activities. In most cases, such as in a corporate environment, the documents listed here will be legitimate, business-oriented documents. However, even in such environments, users may be found accessing documents that they shouldn't. Information about the documents that the user has accessed is maintained in the RecentDocs key, which is found in the following path:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
```

An example of RecentDocs key, as well as the subkeys and values, from a Windows XP (the structure of the data has not changed significantly on Windows 7 and Windows 10 systems) system is illustrated in [Fig. 4.9](#).

As you can see in [Fig. 4.9](#), the RecentDocs key itself contains numbered values (0, 1, etc.) that each contain binary data, as well as subkeys named for the various extensions of the files accessed. Each of these subkeys also contains numbered values with binary data as well. All of these keys contain a value named "MRUListEx",

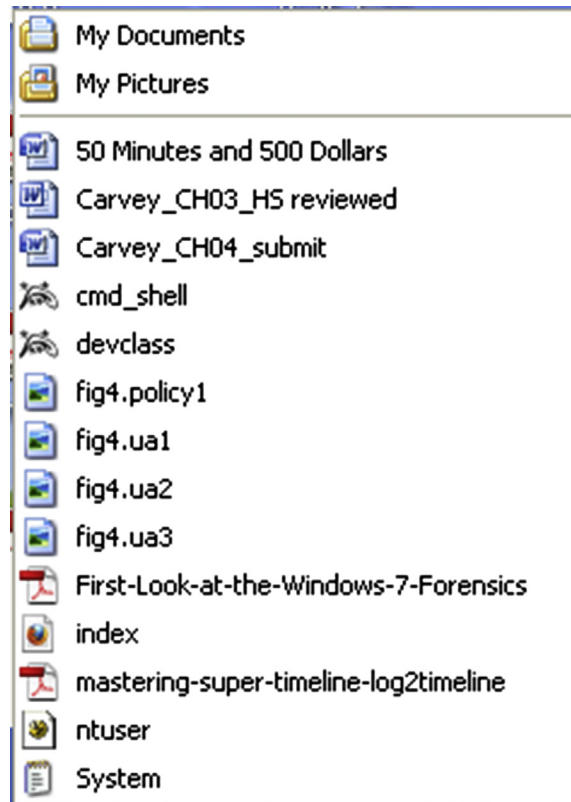


Figure 4.8 Windows XP Recent Documents menu listing.

which is a sequence of DWORD values that list the order in which the documents or files were accessed. Vista and Windows 7 record this information in the same way, and the recentdocs.pl RegRipper plugin can be used to parse the necessary information from the binary value data on all versions of Windows. An example of information retrieved by the recentdocs.pl plugin from a Windows 7 system appears as follows:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\
RecentDocs\.jpeg
  LastWrite Time Sat Mar 13 22:25:46 2010 (UTC)
  MRUListEx = 2,1,0
    2 = anime_155.jpeg
    1 = 11.ca2.jpeg
    0 = roripara22_png.jpeg

Software\Microsoft\Windows\CurrentVersion\Explorer\
RecentDocs\.jpg
  LastWrite Time Tue Mar 16 15:43:58 2010 (UTC)
  MRUListEx = 3,1,2,8,9,4,0,6,5,7
```

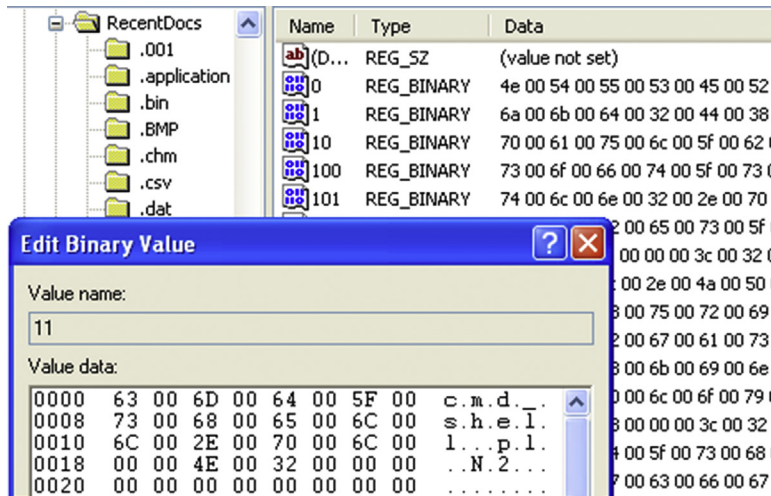


Figure 4.9 View of RecentDocs key/values via RegEdit.exe.

```
3 = Picnik collage.jpg
1 = hether-446.jpg
2 = 09.jpg
8 = 1211720515959.jpg
9 = 016.jpg
4 = 25517_1260411908194_1166566081_30636671_8251529_n.jpg
0 = 25517_1260297105324_1166566081_30636173_6335083_n.jpg
6 = 25517_1260297145325_1166566081_30636174_7038891_n.jpg
5 = 25517_1260297185326_1166566081_30636175_5223984_n.jpg
7 = 25517_1260297225327_1166566081_30636176_4397882_n.jpg
```

This example illustrates the user’s access to .jpeg and .jpg files; in short, images. One thing you’ll notice is that the plugin parses the MRUListEx value and then presents the files in the order in which they are listed in that value. Based on how the contents of these keys are maintained, we can see that anime_155.jpeg was accessed on Saturday, March 13, 2010 at approximately 22:25:46 (UTC), and that “Picnik collage.jpg” was accessed on Tuesday, March 16, 2010 at approximately 15:43:58 (UTC).

WHAT APPLICATION USES OR CREATED THAT FILE?

Many times while I’m perusing online forums, I’ll see a question similar to, “what application is used to access/created this file?” Most of the time, the response is a reference to a Google search (or even a URL for imgtfy.com) or to fileext.com. This may seem like the obvious answer, but it’s not someplace I’d start. When I see a file extension listed on a file in an image, or in the RecentDocs key in the user’s hive, and I’m interested in determining the application that is associated with that file extension

WHAT APPLICATION USES OR CREATED THAT FILE?—Cont'd

on the system, I'll run the *assoc.pl* RegRipper plugin against the Software hive (via *rip.pl/.exe*), redirect the output to a file, and then look to see what may be listed in the output file. This allows me to determine the file associations on that system; searching for this information via Google, while it may be useful, does not address the context of what applications are installed on the system being analyzed. The output of the *assoc.pl* plugin can also tell me about installed applications; for example, on a Windows 7 system, I found that all of the graphics files (.jpg, .img, .tif, etc.) were associated with the IrfanView application. So, not only did I now know that IrfanView was installed, but I now had another application to check for an MRU list of opened or saved files. From this same system, I also found that OpenOffice was installed rather than Microsoft Office. Searching via Google may provide useful leads, but examining artifacts on the system being examined will many times provide much-needed context.

However, this information applies to the system itself; file association settings from the user profile (found in the user's *USRCLASS.DAT* hive) will supersede the system settings when the user logs in. This is covered in more detail in the “[File Associations](#)” section later in this chapter.

We can see from this that the values beneath the *RecentDocs* key and its subkeys will tell us what documents and files the user account was used to access (I say that, because that's all we know...we don't really know who was at the keyboard when the account was logged in...), as well as when the most recently access document was accessed (via the first item in the *MRUListEx* value and the key *LastWrite* time). A closer look at the binary data for the various values shows us the file name and a referenced Windows shortcut (.lnk) file, but not the full path to the file itself, so we don't know if the file was on the local hard drive, on a CD, thumb drive attached to the system, or on a network share.

ComDlg32

The key “*ComDlg32*” refers to common dialogs available on Windows systems. Rather than requiring developers to recreate or code from scratch some of those dialogs that are used frequently, these are actually provided for use through the Windows application programming interface (API). The path to the key is *Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32*, and the keys of interest beneath this key differ slightly between Windows XP and Windows 7 and beyond. [Figs. 4.10 and 4.11](#) illustrate the keys on Windows XP and Vista, respectively.

The *OpenSaveMRU* (on Windows XP and 2003; *OpenSavePid-MRU* on Vista through Windows 10 systems) tracks files that the user account is used to access via the Open and Save As... common dialogs. You can see the use of these common dialogs when

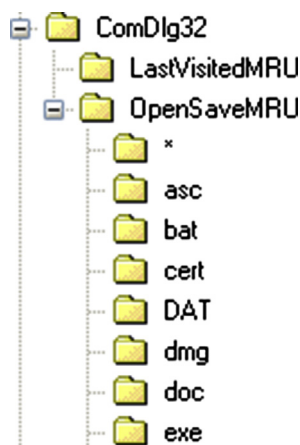


Figure 4.10 Windows XP ComDlg32 key.

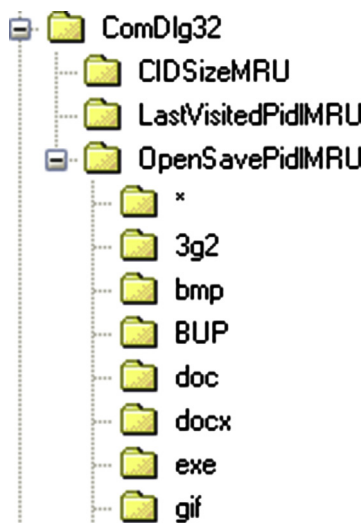


Figure 4.11 Windows Vista ComDlg32 key.

opening an application and clicking on the File menu item. From there, the drop-down menu will include Open and Save As... options, and choosing these options will launch the common dialogs. This key and its subkeys also track previously opened or saved files as an autocomplete feature, as illustrated in [Fig. 4.12](#).

As you can see illustrated in [Figs. 4.10 and 4.11](#), the OpenSaveMRU and OpenSavePidlMRU keys contain subkeys that specify the extensions of the files opened or saved. In [Fig. 4.10](#), we see a subkey named “asc”, which refers to files used by the Pretty Good

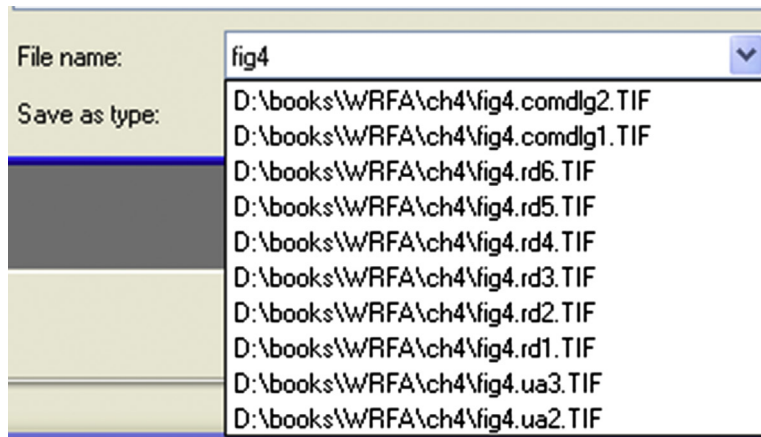


Figure 4.12 MS Paint Save As... dialog autocomplete listing.

Privacy (PGP) encryption application. Each of these keys contains values whose name letters and whose data points to the files in question. Each key also contains an MRUList value which is a string that lists the MRU order in which the files were accessed. As such, the LastWrite time of the key would correspond to the time that the first file referenced in the MRUList value was accessed. The OpenSavePidMRU subkey values are different, in that the values are binary data types and need to be parsed appropriately to retrieve the file name; also, the subkeys each contain a value named MRUListEx (as opposed to a value named MRUList), which is also a binary data type and needs to be parsed appropriately as well.

One subkey beneath the OpenSaveMRU and OpenSavePidMRU keys that stands out is the key named "*". This refers to files of any extension, or no extension, and also maintains the list of most recently accessed files for each type. For example, beneath the OpenSaveMRU key in [Fig. 4.10](#) is a subkey named "zip", which contains six values. The most recently accessed file that ends with the ".zip" extension is not only listed in the MRUList value within that key, but it is also listed as a value in the "*" subkey.

The LastVisitedMRU (LastVisitedPidMRU on Vista and Windows seven systems) key serves a bit of a different function. This key tracks the application last used to access the files listed in the OpenSaveMRU key (and its subkeys), as well as the directory that was last accessed. The OpenSaveMRU values include the paths and file names; also, remember that the common dialogs (in this case Open and Save As...) are not applications in and of themselves but are instead accessed via other applications, such as MS Paint, Notepad, MS Word, the web browser, etc. [Fig. 4.13](#) illustrates a LastVisitedMRU value.

	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F	0123456789ABCDEF
0x00	5000	4F00	5700	4500	5200	5000	4E00	5400	P.O.W.E.R.P.N.T.
0x10	2E00	4500	5800	4500	0000	4300	3A00	5C00	.E.X.E...C...\.
0x20	6400	6F00	6300	7300	5C00	4800	4B00	0000	d.o.c.s.\.H.K...

Figure 4.13 Windows XP LastVisitedMRU value viewed via RFV.

In Fig. 4.13, we see the executable which was used to access the common dialog (Powerpnt.exe) and the directory that it was used to access (C:\docs\HK). Using this information, we can then correlate the values based on the LastVisitedMRU key's MRU-List value to the values found beneath the OpenSaveMRU* key in order to obtain path information. For example, the first value referenced in the LastVisitedMRU MRUList value is "f", which points to Winword.exe, and includes the C:\docs\xcel directory in the binary data. We then go to the OpenSaveMRU* key, and the first value listed in the MRUList value is also "f", which in this case points to C:\docs\xcel\xcel.doc. However, remember that these are MRU keys, so we shouldn't expect to find a great deal of historical data that would allow us to track file paths back several weeks or months.

HISTORICAL DATA

Let's not forget that while some Registry keys (such as the ones that maintain MRU information) can show us not only the most recent documents that a user account had been used to access but also documents accessed in the past, analysts can also find further historical data in Windows XP System Restore Points or within Volume Shadow copies (as on Vista and Windows 7 systems).

Similar to the OpenSavePidMRU key values, the values listed within the LastVisitedPidMRU key (Vista through Windows 10) are binary data types and should be parsed appropriately. However, these values contain similar information as their counterparts on Windows XP and 2003 systems. The *comdlg32.pl* RegRipper plugin will extract and display the information from the values beneath the ComDlg32 key and its subkeys, but the caveat of continued research and input into the maintenance and development of the plugin (and others) to address new data types remains.

Microsoft Office File/Place MRUs

Most Windows systems, particularly those in a corporate environment, have the MS Office suite of products installed.

Thankfully, the MS Office applications maintain their own MRU lists, which are maintained in a path similar to the following:

```
SOFTWARE\Microsoft\Office\version\application name\File MRU
```

For MS Office 2010, the path for the “File MRU” key for Excel is *Software\Microsoft\Office\14.0\Excel\File MRU*. The path is similar for MS Office 2013 (version 15.0) applications, with the version listed as “15.0”.

LIVEID ACCOUNTS

In some test scenarios, I’ve found the *File MRU* key path to be *Software\Microsoft\Office\15.0\Word\User MRU\LiveId_{hash}\File MRU* for the MS Word application that is part of the MS Office 2013 suite. This appears to be related to use of a Microsoft LiveID account and clearly will require additional research.

Something very useful about the values beneath these keys is that they have a time stamp value embedded within the value data for each of the MRU values. In 2011, Cameron Howell shared his code for the *office2010.pl* plugin, which not only parses the file path and name from the data, but it also parses and displays the time stamp. This plugin needs to be updated to include MS Office 2013 documents as well as MRU lists for LiveID accounts.

TrustRecords

For systems that do have MS Office installed, there may be another source of information available. When a user downloads an MS Office document from a network location, or from the Internet, and then opens the document, they see the yellow “Protected View” bar across the top of the document, as illustrated in [Fig. 4.14](#).

The user can read the document even with the Protected View bar visible, but in order to edit (or print) the document, the user needs to click on the “Enable Editing” button. When they do so, an entry is created in the TrustRecords key. For MS Word, the path to the key is:

```
Software\Microsoft\Office\14.0\Word\Security\Trusted Documents\TrustRecords
```

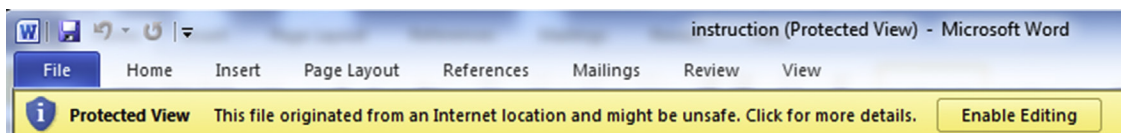


Figure 4.14 Enable Editing bar in MS Word.

Beneath this key is a value for each of the documents for which the user clicked the “Enable Editing” button. The first 8 bytes of the value data are a 64-bit FILETIME object that indicates the date and time that the user clicked on the button to enable editing of the document. As such, each value (ie, path to the document) has its own individual time stamp, as illustrated in Fig. 4.15.

MICROSOFT WORD

The TrustRecords key not only illustrates that the user accessed a particular file, and when, but it also illustrates the execution of the MS Word application.

An 18-page PDF document written by Dustin Hurlbut (available online at https://ad-pdf.s3.amazonaws.com/Microsoft_Office_2007-2010_Registry_ArtifactsFINAL.pdf) provides an extensive list of Registry artifacts associated with MS Office 2007 and 2010 and includes the TrustRecords artifacts.

Adobe Reader

On July 16, 2015, Jason Hale posted to his blog (found online at <http://dfstream.blogspot.com/2015/07/adobe-readers-not-so-recentfiles.html>) detailing new values he’d found in the user’s hive with respect to the Adobe PDF reader application. In short, he’d not only found that the number of previously viewed files was increased (that is, more keys were created and maintained), but also that several new values had been added to each key. Jason indicates in his post that as of version 11.0.07 of Adobe Reader application, there are a total of 100 subkeys that are maintained (up from the previous 5), one for each document the user opened. In addition, several values have been added to each subkey, including one for the page count of the document, as well as one for the size of the file itself. There is also a value named “sDate” that has been added, which appears to indicate the date at time for when the file was accessed. This value has a binary format but has a string format similar to “D:20150511165429-4’00”, which indicates that the file was accessed at 4:54:29 pm on May 11, 2015,




	file:///D:/books/Perl/Perl%20Forensics%20Cookbook_2.docx	REG_BINARY	6A BA 2D 6C 4D EC CC 01 00
	file:///D:/books/Perl/Perl%20Forensics%20Cookbook_4.docx	REG_BINARY	17 E2 A5 46 C9 EF CC 01 00 F
	file:///D:/books/Perl/Perl%20Forensics%20Cookbook_5.docx	REG_BINARY	75 43 E8 4D 04 F1 CC 01 00 F1

Figure 4.15 TrustRecords values.

local time (the “-4” appears to indicate the time zone offset from GMT or UTC).

Why is this important? As the user views successive files, each previously viewed file is “pushed down,” and the most recent one is written to the “c1” key. For example, on a system with a fresh installation of the application, the first file viewed would be written to the “c1” subkey; when the second file is viewed by the user, the information about the first file viewed is written to the “c2” subkey, and the information regarding the most recently viewed file is written to the “c1” subkey, and so on. What this means is that all of the subkeys (up to 100) will have the same LastWrite time, so that information will be of little value in a timeline that is meant to illustrate the user’s activities. However, this new “sDate” value can be translated to a time stamp value, and that information can be used in a timeline.

DATA SOURCES

While data sources outside of the Registry are beyond the scope of this book, I think that it’s important to point out that Jump Lists (available on Windows 7 through 10 systems) serve as excellent sources of MRU data when it comes to user access to files.

So far in this section, we’ve listed some of the locations within the Registry that are used to record a user’s access to files. As with the rest of this book, this list should not be considered complete; there are far too many combinations of applications and versions to provide a complete list. What I’ve attempted to illustrate thus far in this chapter is that the Registry records a good deal of user activity, and in many cases, associates that activity with a time stamp, making the Registry an exceptional resource for forensic analysts.

User Activity

So far we’ve described a great deal of user activity that’s recorded in the NTUSER.DAT file. As one might expect, there’s even more information recorded in the Registry that can be categorized under “user activity.” This category covers other activities that apply to actions the user took but don’t fit into the previous categories.

TypedPaths

The TypedPaths key (*Software\Microsoft\Windows\Current-Version\Explorer\TypedPaths*) records paths that the user typed into Windows Explorer. On Windows 10, typing into the little box to the right of the Windows icon on the TaskBar, the one that usually says, “Search the web and Windows”, will populate this key. The



Value	Type	Data
 url1	REG_SZ	mspaint
 url2	REG_SZ	Control Panel

Figure 4.16 TypedPaths key values.

values within this key have names such as “url1” and “url2”, as illustrated in Fig. 4.16.

When the first entry is typed, the first value to be added to the key will be “url1”. When the next entry is typed, the new value is named “url1”, and previous value becomes “url2”, and so on. As such, under normal circumstances, the LastWrite time of the key will correspond to when the most recent entry was typed.

Again, that’s under normal circumstances and doesn’t apply if the user modifies the key through the Registry Editor or through the use of code, such as a Visual Basic script. However, I have rarely seen this value manipulated in that manner; in fact, I don’t recall a time when I’ve seen the content of this key manipulated specifically to obscure a user’s activity. That does not mean that it could not happen, I just haven’t seen it yet.

TypedURLS

Forensic analysts have long associated the values beneath the TypedURLs key in the user’s NTUSER.DAT hive file with user activity. It’s understood that when a user types in a website address in the Internet Explorer (IE) address bar, that URL is written to the URL1 value beneath the key.

In March 2011, Paul Nichols of Crucial Security posted an article (found online at <http://crucialsecurityblog.harris.com/2011/03/23/typedurls-part-2/>) regarding some potential issues with the accepted belief the values beneath the TypedURLs keys are modified solely by user activity. As the article points out, there is a default value listed beneath the key, and there are several malware variants that modify values beneath the key.

WRITING TO THE REGISTRY

While many Registry keys and values within the user’s hives are modified when a user performs specific actions, it is important to remember that there may be other reasons for the data that you’re seeing beneath specific keys. As has been mentioned several times throughout this chapter and the previous one, there are variants of malware that write directly to the Registry (as opposed to the operating system creating entries based on the presence of the malware), many for purposes other than persistence.

The contents of the TypedURLs key have proved fruitful during breach investigations, particularly when a dedicated adversary has access to the infrastructure via Terminal Services. In several instances, I've found the values beneath this key to be invaluable in determining an intruder's activities, particularly when they were accessing a web shell. The TypedURLs key contains values that appear as follows:

```
url19 -> http://pandora.com/
url20 -> http://192.168.1.1/
url21 -> \\fileserver
```

Much like the TypedPaths key, as the user types new URLs into the IE address bar, the most recently typed URL becomes url1, and each of the previous entries get "pushed down." As such, under normal circumstances, the LastWrite time of the TypedURLs key will correspond to when the user typed the contents of the url1 value into the IE address bar.

Another key that was found when Windows 8 was released and later determined to be associated with IE 10 is the "TypedURL-sTime" key. Under normal circumstances (that is, without outside manipulation), each of the values beneath this key corresponds to value of the same name beneath the TypedURLs key and provides the date and time when the user typed the URL.

Searches

Users will often search for things (files by name, keywords within files, etc.) on their systems, as well as other systems, and on the Internet. Sometimes, they even do this using the built-in search capability that comes with Windows XP, as illustrated in [Fig. 4.17](#).

When a user runs a search on Windows XP, the information about what is being searched for is maintained in the following Registry key:

```
Software\Microsoft\Search Assistant\ACMr
```

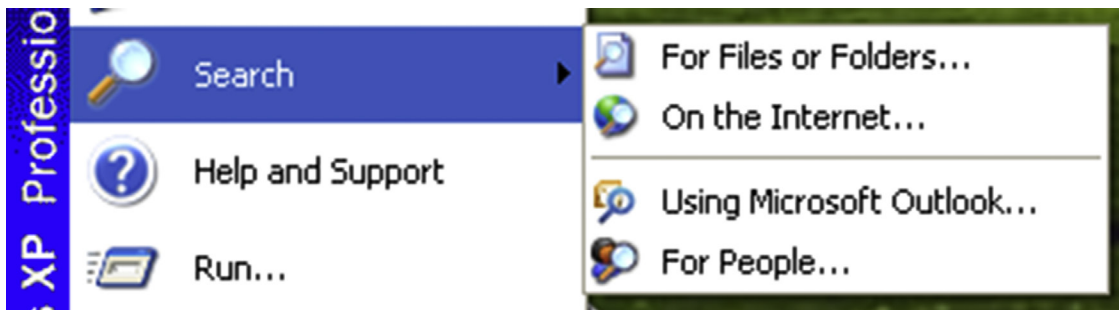


Figure 4.17 Windows XP Search.

Beneath this key are several subkeys, each of which is named for a number, and each of these numbers corresponds to a particular portion of the Search Assistant, as indicated as follows:

- 5001—contains list of terms entered via the “On the Internet...” search
- 5603—contains the list of terms entered via the Windows XP “For Files or Folders...” search
- 5604—contains list of terms searched for using the “A word or phrase in the file” search
- 5647—contains list of terms searched for using the “Computers or people” search

Fig. 4.18 illustrates the portion of the Search Assistant in which entries populate the 5603 and 5604 keys, respectively.

I have found this information has proven to be very useful during a number of examinations. For example, the values beneath these keys are also numbered in an MRU fashion: 000, 001, 002, etc. Therefore, the LastWrite time for the key itself lets us know when the search for the “000” value was conducted. Sometimes I find entries that are entirely normal for a particular

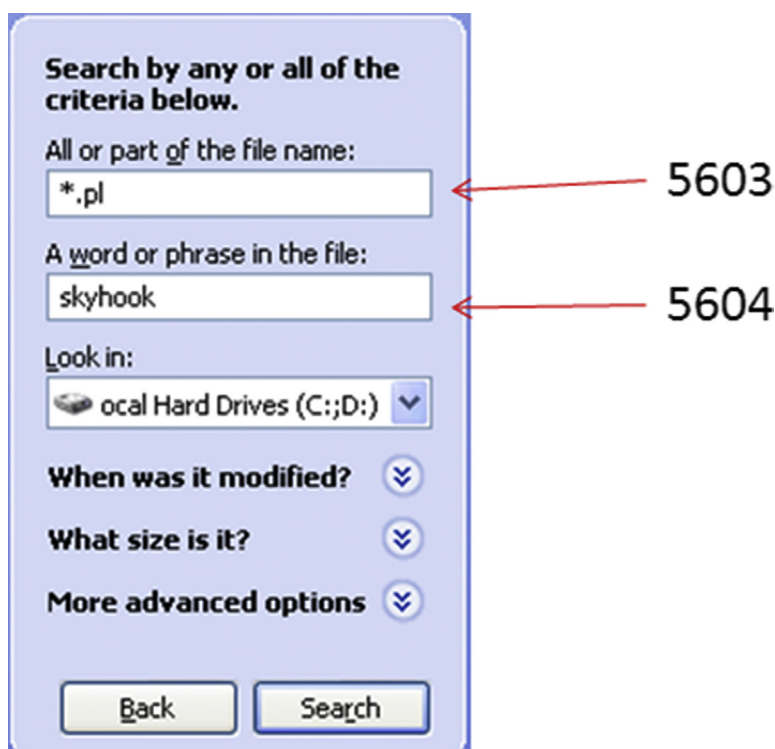


Figure 4.18 Windows XP Search Assistant to ACMRu subkey mappings.

user; in other cases, perhaps not so much. For example, I've seen where someone who had no business doing so was searching for terms such as "banking" and "passwords." I've also seen where someone has perhaps had trouble spelling, searching for "bankign."

On Vista systems, information about searches run by the user is maintained in a file and not within a Registry key. With Windows 7, information about what the user searched for is again recorded in the Registry, this time in the WordWheelQuery key. The full path to this key appears as follows:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\
WordWheelQuery
```

Fig. 4.19 illustrates how the contents of this key appear in Windows Explorer on Windows 7.

The values within the WordWheelQuery key are binary data types that are numbered ("0", "1", etc.), and there is also an MRUListEx value that is also a binary data type. As with many MRUListEx values, the MRU list is maintained as 4-byte DWORD values in sequence, with the value 0xFFFF indicating the end of the list. As with the Windows XP ACMru key, the information in this key may shed some light as to the user's activity on the system.

The traditional approach to computer forensic analysis has relied heavily on file system time stamps and a few other artifacts

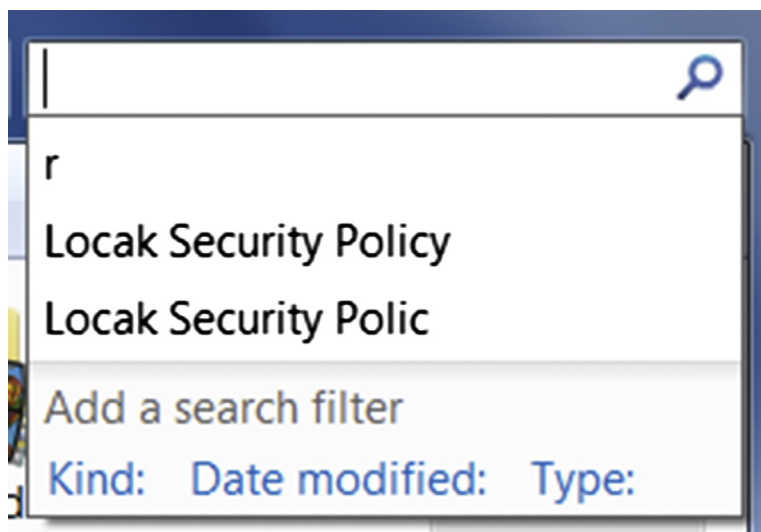


Figure 4.19 Windows 7 Search history.

(file contents) found on the system. However, systems are now often accessed via multiple user accounts, and the scope of many investigations has expanded beyond the boundaries and hard drive of just one system. Further, there are times when an analyst cannot trust file system time stamps, as either the updating of file last access times is disabled, or an intruder (or malware) modified those time stamps. As such, we need to look to other locations within the system to develop a better understanding of activity associated with a user account. The best place to start is within the Registry hive files within the user profile; there is the well-known NTUSER.DAT hive found in the root of the profile directory, and with more recent versions of Windows (Vista, Windows 7), the USRCLASS.DAT hive is seeing greater usage. In this chapter, we'll focus on discussing and demonstrating how activity associated with a user account ("user activity") is recorded in the user's hives, and how analysts can use that information to the benefit of their examinations.

File Associations

We discussed in chapter "[Analyzing the System Hives](#)" how file association information from within the Software hive can be used to answer questions regarding the relationship between file extensions and applications on a system. However, information about file associations is also maintained on a per-user basis, as well. If you open the Registry Editor on a live system to the HKEY_CURRENT_USER hive and expand the tree beneath the Software key, you'll see a Classes subkey with information similar to that which appears in the Software hive. This information is mapped into the HKEY_CURRENT_USER hive from the user's USRCLASS.DAT hive file and supersedes information available in the Software hive.

As an example of this, we can start with the discussion of the default web browser from the "Web Browser" section of chapter "[Analyzing the System Hives](#)". In that particular case, the information from the Software hive indicates that Internet Explorer is the default web browser for the system. However, the "Default" value from the following key (from the live system, accessed via regedit.exe) points to the Firefox web browser:

```
HKEY_CURRENT_USER\Software\Classes\http\shell\open\command
```

This maps to the following key found in the USRCLASS.DAT hive within my user profile:

```
http\shell\open\command
```

What this shows is that when I log into the system with my account, the system settings for file associations (and in this particular case, the default web browser) are superseded by settings found in my USRCLASS.DAT hive.

Note

As we've seen, there are a number of instances where Registry artifacts that indicate the installation or use of applications persist after the application is removed or deleted. This applies to many applications that simply have a GUI but do not require an installation routine (ie, the application files are simply copied to a directory). However, many applications that utilize an installation routine and set file associations in the Registry will also "undo" those settings when the application is uninstalled. This is yet another example of how Registry hives from System Restore Points (Windows XP) or Volume Shadow Copies, as well as deleted keys extracted from unallocated space within Registry hive files (via regslack.exe), can provide significant historical data from a system.

USRCLASS.DAT

Throughout the evolution of Windows systems, one of the things I've noted is that more and more has been "moved" to the USRCLASS.DAT hive. That's not to say that everything has...not at all. One example is the shellbags artifacts discussed later in this chapter; these artifacts were found in the NTUSER.DAT hive with Windows XP and 2003 systems and were "moved" to the USRCLASS.DAT hive in Vista systems, where they have remained. With the release of Windows 8 (and subsequently, Windows 8.1 and Windows 10), there appears to be the potential for even more data that may be relevant to investigators stored within this hive; I'm sure that as Windows 10 systems become more popular and in wider use, that data will be documented.

AutoStart

Within the user's USRCLASS.DAT hive, the location where we most often seen used to autostart programs (and subsequently, malware) is the *InProcServer32* key beneath the *CLSID* key; the full path within the hive is *CLSID\{GUID}\InProcServer32*, as illustrated in [Fig. 4.20](#).

As with other autostart locations, this key path is most often used by legitimate applications but is also hijacked by malware. One such example is the malware downloader known as "Lurk," which was discussed in an article posted to the Dell SecureWorks

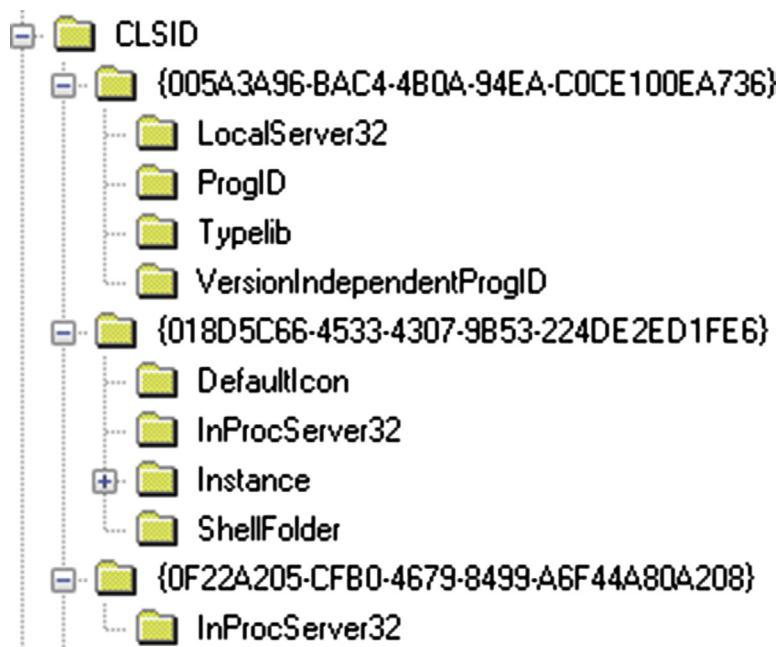


Figure 4.20 Partial contents of Windows 10 CLSID key.

website in November 2014 (found online at <http://www.secureworks.com/cyber-threat-intelligence/threats/malware-analysis-of-the-lurk-downloader/>). According to the article, the Lurk malware persists by hijacking the “(Default)” value beneath the `CLSID\{A3CCEDF7-2DE2-11D0-86F4-00A0C913F750}\InProcServer32` key and pointing to the malware DLL rather than the Internet Explorer PNG image decoder. Performing a search via Google for the GUID shows us other malware that has been seen to use this same key path for persistence.

The RegRipper *inprocserver.pl* plugin can be used to search across the Software, NTUSER.DAT, and USRCLASS.DAT hive files for anomalous entries. If you open the plugin in an editor (Notepad, Notepad++, etc.) and look at the headers, you’ll see that over time I’ve updated and modified the plugin; this is something that I’m sure will need to be done in the future as well, as our understanding of Windows 8, 8.1, and 10 progresses through new research and discovery.

Program Execution

There do not appear at this point to be a great many keys or values within the USRCLASS.DAT hive that contain information

that would be associated with the “program execution” category. The MuiCache key is one of those Registry keys that seems as if it might be very useful, but for which there is very little documentation available. On Windows XP and 2003 systems, the path to the MuiCache key within the user’s NTUSER.DAT hive is:

```
Software\Microsoft\Windows\ShellNoRoam\MuiCache
```

On Vista systems and above, the key path is located in the user’s USRCLASS.DAT hive, in the key path:

```
Local Settings\Software\Microsoft\Windows\Shell\MuiCache
```

So, how is this key useful? Several years ago, I was doing some research on specific malware samples and looking to see what some of the AV vendors had already documented with respect to the variants they’d seen. In some instances, I began to see references to malware creating a value (according to the AV vendor write-up) within the MuiCache key when run, and not being familiar with this key, I wanted to see if I could determine the reason for this value being created. As it later turned out, the malware wasn’t creating the value...the value was being created by the operating system, as a result of how the malware was being launched within the testing environment. This proved to be very interesting and very useful.

We’ve already seen how we can track the user’s activity on a system when they interact with the shell, whether they’re conducting searches or launching applications. However, in some instances, we’ll see that a command prompt was launched (as indicated by the UserAssist key or RunMRU entries) and then nothing afterward. In some instances, we may be able to get an idea of what the user may have done (or more correctly, what the user account may have been used to do...) by examining the contents of the MuiCache key. By default, when an account is first created (or shortly after it is first used) the MuiCache key may contain value names that start with “@”. However, once the profile begins to be used, there may be additional entries that appear as illustrated in [Fig. 4.21](#).

As you can see in [Fig. 4.21](#), this key provides a sort of historic, persistent record of the applications that the user account has




Value	Type	Data
 LangID	REG_BINARY	09 04
 C:\Windows\system32\WFS.exe	REG_SZ	Microsoft Windows Fax and Scan
 C:\Users\john\AppData\Local\Google...	REG_SZ	Google Chrome

Figure 4.21 MuiCache key contents from a Windows 7 system.

been used to run, albeit without any sort of time stamp specific to each application. While conducting forensic analysis during an incident response engagement a number of years ago, I was parsing the NTUSER.DAT file from a compromised Windows 2003 system (using RegRipper's *muicache.pl* plugin), when I noticed that there were several unusual value names that referenced non-native executable files in the "C:\Windows\Tasks" directory. It appeared that the intruder was placing his toolset in this directory, as by default, when viewing the Tasks directory via the Windows Explorer shell on a live system (which is how most system administrators tend to do so), the .exe files do not appear in the viewing pane. This means that the intruder's tools are effectively hidden from view from most of the likely first responders, should any unusual activity be detected on the compromised system. It turned out that we were able to locate several of the tools in the Tasks directory, but several others had apparently been deleted. This provided an interesting indication of the intruder's other activities on the system (ie, they'd apparently added, used/run, and then deleted other command line tools) that remained persistent after the intruder had apparently deleted several of the tools used.

WINDOWS 10 SYSTEMS

During the course of writing this book, I was reviewing data in a USR-CLASS.DAT file extracted from a Windows 10 system and noticed that there wasn't an MuiCache key in the "normal" location; however, I did see two additional keys, one at the path *Local Settings\ImmutableMuiCache*, and the other at *Local Settings\MuiCache*. I didn't see a great deal of particularly useful information beneath either of these keys, but that may have been because the profile hadn't been used to any great extent. That would also perhaps explain why the "normal" MuiCache key did not appear to be present and simply reinforces that additional and continual research is required in the digital forensic analysis field.

Several years ago, I used to present pretty regularly at local High Tech Crime Investigation Association (HTCIA) conferences (our local chapter became known as the Regional Computer Forensics Group, or RCFG) and spoke to a number of law enforcement officers about the issue of steganography or hiding programs or files inside other files. While steganography was mentioned in the media, as well as within a number of training courses, I was curious as to how prevalent it was seen within the law enforcement community. Interestingly enough, not one of the law enforcement officers I spoke to could recount ever having seen or suspected the use of steganography in any of their examinations. While there

are a number of freely available tools for embedding or hiding files (executable files, images, text, videos, etc.) within other file, many of them do not get installed on a system in the usual sense; instead, the application files are simply added to a directory by the user. The contents of the MuiCache key may indicate the use of steganography applications, particularly those that may have been copied to a system or run from external media, such as a CD or thumb drive.

Overall, the point of this is that, under most normal circumstances, values beneath the MuiCache key generally appear as a result of interaction of some kind with the shell. When an executable file path is found as a value name beneath this key, it appears to indicate that the user account in question was used to run the application. Follow-on analysis steps might be to attempt to locate the file within the file system (or unallocated space), a Prefetch file, or perhaps an MFT entry (particularly if the file path indicates that the file was on a local hard drive). This key can provide some very interesting indications of activities that occurred within the context of the user account.

File Access

The majority of locations within the user's hive files that provide indications of file access are found within the NTUSER.DAT, but there are some key paths within the USRCLASS.DAT hive that provide indications of the user accessing certain types of files.

Photos

In March 2013, Jason Hale posted to his blog (found online at <http://dfstream.blogspot.com/2013/03/windows-8-tracking-opened-photos.html>) regarding functionality inherent to Windows 8; specifically, when a user double-clicked an image file, it would be opened via the Photos application (or "tile") on what was referred to as the "Metro" desktop. Information regarding these files is buried deep within the user's USRCLASS.DAT hive file, including (but not limited to) the full path to the opened image file and a value named "Link" whose binary data is a variation on the Windows shortcut format. Yes, that's exactly what it sounds like; a value buried within the hive file contains what amounts to Windows shortcut, or *.lnk, formatted data.

Interestingly enough, Jason updated the blog post over a year later, stating that Windows 8.1 systems do not appear to record or maintain this data. This artifact is obviously different from the file access MRUs discussed previously in this chapter, but I wanted to include it here in case someone reading this book has to perform

forensic analysis of a Windows 8 (not 8.1) system and is interesting in knowing which image files may have been accessed by a user.

Tip

This is yet another example of why knowing the version of Windows you're analyzing is important, and how sharing that information when you ask questions in online forums can prove fruitful.

Shellbags

As has been discussed thus far, with respect to versions of Windows beyond XP, a good bit of functionality has been added to the operating systems, functionality that makes Windows much more of a “user experience.” As a result, some of the information recorded in order to enable the functionality was moved to the USRCLASS.DAT hive.

Perhaps the most notable artifact within the USRCLASS.DAT hive is referred to as *shellbags*. This artifact is referred to as “shellbags” due to the name of the one of the Registry keys involved; on Windows 7 through Windows 10 systems, the path to the artifacts is:

```
Local Settings\Software\Microsoft\Windows\Shell\BagMRU
```

Note

Again, with Windows XP and 2003 systems, the shellbags artifacts were maintained in the NTUSER.DAT hive, and you could even get directory listings of files from the Registry values, in some cases. That all changed when Windows Vista was released, and pretty much the only thing that's changed since then is that over time, new shell items have been released. As such, continual research in this area has been required, and Eric Zimmerman has put forth considerable effort into keeping abreast of new developments and artifacts in this area.

Once again...the version of Windows you're analyzing can play a significant role in determining what artifacts are available, and where those artifacts are located (within the file system, Registry, etc.).

Beneath this key path is a series of nested subkeys, as illustrated in [Fig. 4.22](#).

Beneath the BagMRU key, and beneath the nested subkeys shown in [Fig. 4.22](#), you'll find “MRUListEx” values along with numbered values (ie, 0, 1, 2, 3, etc.), as illustrated in [Fig. 4.23](#).

As discussed previously in this chapter, the MRUListEx value simply indicates the order in which each of the numbered values

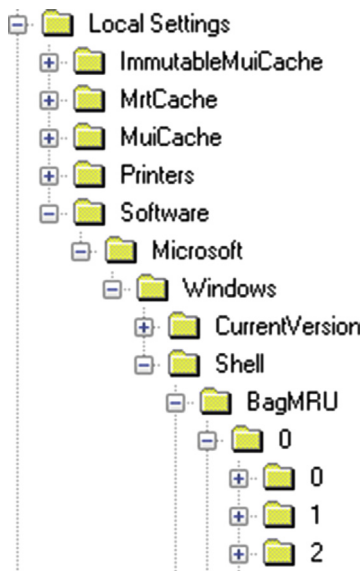


Figure 4.22 Windows 10 shellbags artifacts (via WRR).










Value	Type	Data
 NodeSlot	REG_DWORD	0x00000001
 MRUListEx	REG_BINARY	06 00 00 00 04 00 00 00 02 00 00 00 01 00 00
 0	REG_BINARY	0C 00 01 00 84 21 DE 39 01 00 00 00 00 00
 1	REG_BINARY	0C 00 01 00 84 21 DE 39 03 00 00 00 00 00
 2	REG_BINARY	0C 00 01 00 84 21 DE 39 02 00 00 00 00 00
 3	REG_BINARY	0C 00 01 00 84 21 DE 39 09 00 00 00 00 00
 4	REG_BINARY	0C 00 01 00 84 21 DE 39 05 00 00 00 00 00
 5	REG_BINARY	0C 00 01 00 84 21 DE 39 08 00 00 00 00 00
 6	REG_BINARY	0C 00 01 00 84 21 DE 39 00 00 00 00 00 00

Figure 4.23 Description.

was accessed. For example, in [Fig. 4.23](#), the first DWORD (ie, 4 byte) segment of the MRUListEx value data is “6”, indicating that the item in the value named “6” was most recently accessed.

Now, the data associated with each of the numbered values is what’s referred to as a “shell item.” This is essentially a “blob” of binary data that contains considerable information about a specific object or element that is part of a complete path.

Tip

These “shell items” are used elsewhere on Windows systems, not only within the Registry (associated with other artifacts) but also within the file system. The most notable artifact within the file system that is made up of shell items, at least in part, is Windows shortcut or “LNK” files.

Depending on the type of data and where we are in the overall path, each of the data blobs can contain various bits of information about the item or object in question. For example, some shell items contain simply a GUID, and we can do a lookup of that GUID to determine if it refers to a special system folder, to the Control Panel, etc. Other shell items may refer to folders on the system, and the data blob may contain a name (ie, “system32”), in addition to file system time stamps for the folder (at the time that the object was accessed by the user), as well as the MFT reference number for that object. One of the extremely difficult aspects of parsing shell items is that doing so requires time and effort, as Microsoft doesn’t provide a great deal of documentation for these data objects.

Parsing through the nested Registry keys, we can reassemble paths based on our understanding of the contents of the individual shell items. For example, we can parse and reassemble the shell items into a path to a folder (ie, “My Computer\D:\Tools\ timeline”), to a zipped archive (ie, “My Computer\D:\Tools\ md5deep-4.1.zip\md5deep-4.1”; zipped archives are treated as folders on Windows Vista and above systems), to Control Panel items (ie, “Control Panel\User Accounts\CLSID_User Accounts\ Change Your Picture”), or to devices (such as “My Computer\ Canon EOS DIGITAL REBEL XT\CF\DCIM\334CANON” or “My Computer\DROID2\Removable Storage\dcim\Camera”).

Tip

A great example of the differences between accessing a system via the command line interface (CLI) and via the GUI and the artifacts that are left through different interactions can be seen in a post at the Binary Zone blog from January 2015 (found line at <http://www.binary-zone.com/2015/01/07/forensic-analysis-creating-user-gui-vs-cli>).

Shellbags can also provide indications of program execution. How is that? Years ago, when I first started writing manuscripts for books, my publisher at the time had a methodology for transferring

large files back and forth; rather than using email, they preferred the file transfer protocol, or “FTP.” However, rather than operating at the command line (which most authors likely tend to *not* do), the publisher provided instructions for accessing FTP via the Windows Explorer shell. I read through the instructions a couple of times before attempting the process, and once I had transferred the files, I didn’t think much more about it. Jump forward several years, and I was examining an image of a Windows server that had been accessed remotely via Terminal Services; yes, the password had been relatively trivial and easy to guess. My analysis indicated that the system had, in fact, been accessed several times, likely by different intruders. I could see where one intruder had used Internet Explorer to download and install the Firefox web browser and then used that browser to download other tools.

One of the goals of the analysis was to look indications of data exfiltration, so I began by looking for indications of access to various tools. In examining the Registry hives from the user profiles on the system, I parsed the shellbag artifacts and found some unusual entries that indicated that one intruder had accessed FTP servers via the Windows Explorer shell. Remembering the instructions from my publisher, I followed them again and connected to an FTP server that I know about...and found that doing so produced artifacts just like what I was seeing in my examination of the compromised Windows server!

While the shellbag artifacts do not explicitly illustrate access to files on Windows Vista through Windows 10 systems, they can provide indications of the user accessing folders, Control Panel applets, as well as external storage devices, including external hard drives, smartphones, and digital cameras. The shellbag artifacts can also show that a user accessed zipped archives, well after the archive itself has been deleted from the system, and they can also provide indications that the user accessed FTP sites via Windows Explorer. As with other artifacts, research into these artifacts is, and needs to be ongoing and continuous. Joachim Metz has compiled documentation regarding the format of various shell items and made that format specification available online, in PDF format, at <https://docs.google.com/file/d/0B-VYGsDJPtZIVDNJQ3p-WX0M1b1k/edit>. Willi Ballenthin provides a good description of shellbag artifacts via <http://www.williballenthin.com/forensics/shellbags/>. However, we’re still discovering new things, as new versions of Windows and new applications are released. While I’m writing this chapter, there is no doubt in my mind that there are new shell items available in Windows 10, ones that we’ll become familiar with as this version of Windows becomes more prominent and in more widespread use.

I use the RegRipper *shellbags.pl* plugin to parse shellbags artifacts, and I do try to keep up with new shell items as they appear. Eric Zimmerman has been very helpful in this regard, not only in providing example data but also in sharing his ShellBag Explorer tool, which he described in a blog post which can be found at <http://binaryforay.blogspot.com/2015/05/shellbags-explorer-061-released.html>. Eric also provides copies of his tools for download via the web page <http://binaryforay.blogspot.com/p/software.html>.

BOOK CONTEST

Prior to actually writing this book, I wrote a post on my blog announcing a contest (the post can be found online at <http://windowsir.blogspot.com/2014/10/wrf-2e-contest.html>) in which I solicited stories from analysts as to how Registry analysis helped them during an investigation. I offered up a free copy of the book once it was published to anyone who submitted their story, and that story was used in the book. By the time the contest clock had run out, I had received only a single submission from an analyst who chose to remain anonymous. His story was that during an illicit images case examination of a Windows XP SP3 system, he had used the Shellbags entries (for Windows XP, those are found in the NTUSER.DAT hive file) to illustrate access to specific image files as well as knowledge of the directory contents. Additional information was used (cell phone records and call logs, work schedules, etc.) to determine which individual in the case had accessed the files in question.

While this is a summary of what was submitted, it does illustrate how Registry information, particularly those entries with time stamps, can be used to great effect and can have a significant impact on the analysis.

Summary

Hopefully, what I've been able to illustrate in this chapter is that there is a great deal of information in the user's hives (NTUSER.DAT, and on Vista and above systems, USRCLASS.DAT) that will provide indications of not only what the user did but also when they did it. This can help demonstrate that a system was in use during a specific period; for example, the creation date and last modification time of the NTUSER.DAT file will provide indications of when the user account was first used to log into the system and when it last logged out, respectively, but information from many of the keys (including key LastWrite times and data derived from binary and string values) will provide indications of actions the user took and when they took them. An analyst can use all of this information to develop an understanding of and add context to

other activity found on the system. As with the other hive files, analysis of the user's hives can also assist in determining if the system was infected with malware, or if the user (or an intruder) was responsible for the observed activity.

By now, through these four chapters, I hope that I've been able to illustrate the immense value of Registry analysis. Over the years, I've tracked user and intruder activity, provided information to obviate the "Trojan Defense," and even exonerated falsely accused employees, all by incorporating Registry analysis in my overall examination.