Program Execution Analysis using UserAssist Key in Modern Windows

Bhupendra Singh and Upasna Singh
Department of Computer Science and Engineering, Defence Institute of Advanced Technology (DU), 411025, Pune, Maharashtra, India

Keywords: UserAssist, Windows Registry Forensics, User Activity Analysis, Program Execution Analysis, Malware Analysis.

Abstract: The construction of user activity timeline related to digital incident being investigated is part of most of the forensic investigations. Sometimes, it is desirable to know the programs executed on a system, and more importantly, when and from where these programs were launched. Program execution analysis is very meaningful effort both for forensic and malware analysts. The UserAssist key, a part of Microsoft Windows registry, records the information related to programs run by a user on a Windows system. This paper seeks thorough investigation of UserAssist key, as a resource for program execution analysis. In this paper, the binary structure of UserAssist key in modern Windows (Windows 7/8/10) is presented and compared with that in older versions of Windows (e.g., Windows XP). Several experiments were carried out to record the behavior of UserAssist key when programs were executed from various sources, such as USB device, Windows store and shared network. These artifacts were found to persist even after the applications have been uninstalled/deleted from the system. In the area of program execution analysis, the paper highlights the forensic capability of UserAssist key and compares it with that from similar sources, such as IconCache.db, SRUDB.dat, Prefetch, Amcache.hve and Shortcut (.lnk) files, in order to summarize what information can and cannot be determined from these sources.

1 INTRODUCTION

In digital forensic investigations, it is sometimes essential to carry out program execution analysis on a suspected system to know recently run programs and their execution history. Program execution analysis is helpful in detection of anti-forensic tools executed to thwart forensic examinations. Malware programs have to run before hiding or sending data remotely, thus, program execution analysis may be an important asset to identify them. The UserAssist key, a part of Windows registry, is a very useful resource in the area of program execution analysis to analyze what programs were recently run and their executions history (Carvey, 2005), (Carvey, 2011). The information obtained from UserAssist key can also be leveraged in construction of forensic timeline of user activities.

Program execution analysis and their usefulness in forensic investigations have been widely discussed in various literature published. The evidence of running programs on a Windows system can be found in numerous log files including Prefetch, Pagefile.sys, Event logs, NTFS USN journal, Registry keys, Jump Lists etc. (Eterovic-Soric et al., 2017). The patterns of anti-forensic tools designed to delete artifacts or prevent artifacts can also be determined using the NTFS USN change journal (Lees, 2013) (Talebi et al., 2015). The authors in (Singh and Singh, 2016a) have shown the traces of recently accessed files and run applications in Jump Lists on Windows 10 system. They demonstrated that the traces of run applications persist even after the applications are removed from the system. In (Wong, 2007), the author has described that numerous artifacts related to run programs on Windows system are recorded in UserAssist registry key. According to (Carvey, 2016), not all applications create their existence in the Windows registry. For instance, some peer-to-peer applications do not rely on registry to store information, instead, they use configurations files due to the fact that these applications are cross-platform and Java-based. The authors in (Mee and Jones, 2005), (Carvey and Altheide, 2005), and (Mee et al., 2006) have investigated the UserAssist key to identify the external devices that have been connected to the system in order to run a malware program or an anti-forensic tool. In (Collie, 2013), the author has examined the evidential potential of Windows IconCache.db file and found that

Singh, B. and Singh, U.
Program Execution Analysis using UserAssist Key in Modern Windows.
DOI: 10.5220/0006416704200429
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 420-429
ISBN: 978-989-758-259-2

IconCache.db file stores artifacts related to executable files when they are present on or run from USB connectable devices. Further, The author in (Lee and Lee, 2014) identified the structure and highlighted the importance of IconCache.db file in forensic investigations. The authors demonstrated that IconCache.db file stores full file paths for executable files that have been run, viewed, or installed on a Windows system. In (Khatri, 2015) the author examined the SRUDB.dat file in Windows 8, as a new database for tracking user activities and linking launched programs by users. The author has demonstrated the usefulness of SRUDB.dat database in forensic investigations by tracking Windows store app runs, tracking processes run from external media devices, tracing deleted processes and anti-forensic tools. In (Singh and Singh, 2016b), the authors demonstrated the traces of run applications in Windows Amcache.hve registry hive and their utilization in forensic investigations.

2 UserAssist KEY IN WINDOWS

Microsoft Windows OS implements various logging and error reporting mechanisms to have a record of how system is being used. This information is logged at numerous hidden locations on hard drive to make OS more interactive and user-friendly. These logs can serve as forensic "nuggets" to analysts to exploit. The UserAssist key, a part of Windows registry, was first introduced in Windows NT4 and continued in later versions of Windows including Windows 10. The UserAssist key is maintained by Microsoft in each user's NTUSER.DAT hive file at the following file path (on all versions of Windows): *Software\Microsoft\Windows\CurrentVersion \Explorer\UserAssist* or, on live computer system, at *\HKCU\Software\Microsoft \Windows\CurrentVersion\Explorer \UserAssist*.

Beneath the UserAssist key, there are atleast two subkeys (same across platform, but may be more in different version of Windows) which represent globally unique identifiers (GUIDs). Each GUID key contains a subkey called 'Count' under which the actual values are stored in obfuscated manner (Stevens, 2010). There is no available Microsoft document detailing inside the method and the purpose of obfuscating the values inside *Count* subkey; however, researchers (AccessData, 2008) in forensic community have identified that these values can be decoded into readable format using ROT-13 algorithm. There are publicly available script and GUI tools (Stevens, 2006) (Carvey, 2013) (Zimmerman, 2015) to parse the UserAssist key. The UserAssist key, its subkeys, and their analysis in Windows XP and Windows 7 have been widely discussed in (AccessData, 2008) (Pullega, 2013).

The next subsections describe about the GUIDs and their meaning, followed by the comparison of binary data structure of UserAssist key in different versions of Windows.

2.1 Windows XP and Vista

In Windows XP and Vista, UserAssist keys binary data contains information for the applications launched by a user only via Windows Explorer. In these versions of Windows, following GUIDs key were observed beneath UserAssist key which are common across the same platform:

- {0D6D4F41-2994-4BA0-8FEF-620E43CD2812}- A key that appears to be specific to Internet Explorer 7 (IE7).
- {5E6AB780-7743-11CF-A12B-00AA004AE837}- IE favorites and other IE toolbar objects.
- {75048700-EF1F-11D0-9888-006097DEACF9}- A list of applications, files, links, and other objects that have been accessed.

Below is the list of few frequently used UEME-strings in Windows XP (Pullega, 2013):

- *UEME_CTLSESSION: This entry is for the session ID, it does not hold data about executed programs.*
- *UEME_UIQCUT: This entry counts the programs launched via Quick launch menu shortcut.*
- *UEME_UISCUT: This entry counts the programs launched via Desktop shortcut.*
- *UEME_RUNCPL: This entry keeps data about executed control applets (.cpl).*
- *UEME_RUNPATH: This entry keeps data about executed programs.*
- *UEME_RUNPIDL: This entry keeps data about executed PIDLs.*
- *UEME_UITOOLBAR: This entry keeps data about clicks on the Windows Explorer Toolbar buttons.*

In Windows XP and Vista, it is easier to determine how an application was launched. For instance, the path information for *cmd.exe* when decoded using ROT-13 method, was recorded under UserAssist key value as *UEME_RUNPATH: C\Windows\system32\cmd.exe*.

2.2 Windows 7 and Higher Versions

On a Windows 7 system, only two GUID keys (different from Windows XP) were observed as below:

1. {CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}- A list of applications, files, links, and other objects that have been accessed.
2. {F4E57C4B-2036-45F0-A9AB-443BCFE33D9F}- Lists the shortcut links used to start programs.

However, on Windows 8 and higher versions, nine GUID keys including above two keys, were observed under UserAssist key. The UEME-strings are reduced in Windows Vista and further reduced in Windows 7 and higher versions. Only two UEME-strings: *UEME_CTLCUACount:ctor* and *UEME_CTLSESSION* are remained in Windows 7-10.

2.3 Structure of Binary Data

The most significant change of UserAssist keys in Windows 7 was a new format for binary data. In Windows 7, the size of the binary data was observed to be 72 bytes as opposed to 16 bytes in Windows XP/Vista (Stevens, 2010). This structure appears to be same in modern Windows (Windows 7-10). Figure 1 illustrates the binary structure of a UserAssist key entry in Windows 10 Pro v1607 system and compares it with that in Windows XP (refer to, Table 1). The newly added fields can significantly effect the investigations, as the new binary structure contains a great deal of information. The first four bytes identifies a session that appears to vary across same platform. The next four bytes starting at offset 4, represents a counter that increments upon program execution. Thus, this attribute can supply background information on the frequency applications have been executed on the system. For a certain application, this value differs from corresponding Prefetch file run count, as UserAssist key is user-specific while Prefetch is not. On a Windows XP/Vista machine, the run count starts with the number '5' as default value. So if we are manually parsing this data, remember to subtract 5 from the run counter when dealing with these machines. The new structure can reveal the focus time (in milliseconds) an application had. This can provide information of how long a program (benign or evil) was being executed on the system by a particular user. The last execution timestamp (in FILETIME) when decoded can provide the date and time when the application was last time executed.

In modern Windows, the UserAssist key also keeps track of applications how they were launched. For instance, the path information for *cmd.exe* when decoded using ROT-13 method, is recorded under UserAssist key value as *1AC14E77-02E7-4E5DB744-2EB1AE5198B7*\cmd.exe. Unlike in Windows XP, *UEME_RUNPATH:* strings appeared to be missing in modern Windows. However, new GUIDs were introduced to keep track of applications how they were launched. *1AC14E77-02E7-4E5DB744-2EB1AE5198B7* GUID corresponds to *C:\Windows\System32*. Table 2 presents the List of known folders and their associated GUIDs in modern Windows.

3 METHODOLOGY

The following subsections describe the objective of the study, research methodology, tools, and the experiments formulated in accordance with the objective.

3.1 Objective

The objective of this study is to investigate the behavior of the UserAssist key, as a result of installing or launching applications from various possible ways. It will also examine the artifacts being created and retained in UserAssist key when application are executed from Window store (in Windows 8+) and shared network. The effects of anti-forensic activities on UserAssist key are also taken into consideration.

3.2 Research Methodology

The research in this article was approached through observation and experimentation. For analyzing the behavior of UserAssist key, several experiments were conducted on six test computers (three hosts and three virtual machines) running different versions of Windows system including Windows 7 Professional SP1, Windows 8 Pro and Windows 10 Pro v1607. The clean environment for each virtual machine was created using VMWare Workstation v11.1.0. NTUSER.DAT file from each virtual machine was extracted from virtual hard disk using FTK imager while the same from each host machine was extracted using a live boot USB of Ubuntu 16.04 LTS following the system shutdown. This is authors opinion that this is the best method to analyze the behavior of UserAssist key due to the fact that installing (and hence for analysis) any forensic tool, such as FTK, Encase etc. would modify the content of UserAssist key.

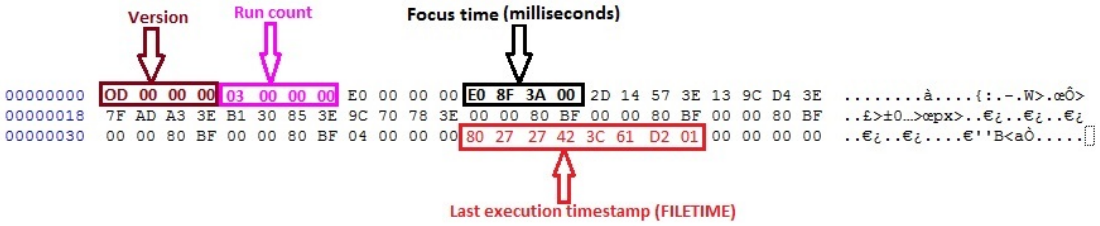


Figure 1: Binary data structure of a UserAssist key entry in Windows 10 Pro v1607 system.

Table 1: Comparison of UserAssist key binary data structure in older and modern Windows.

Binary data structure in Windows XP			Binary data structure in Windows 7-10		
Offset	Size (bytes)	Description	Offset	Size (bytes)	Description
0-3	4	Identifies a session	0-3	4	Identifies a session
4-7	4	Run count (starts at 5)	4-7	4	Run counter (starts at zero)
8-15	8	Last execution timestamp (in FILETIME)	8-11	4	Focus count
			12-15	4	Total time an application had focus (milliseconds)
			16-59	44	Unknown
			60-67	8	Last execution timestamp (in FILETIME)
			68-72	4	Always ‘0x00000000’

Table 2: List of known folders and their associated GUIDs in modern Windows.

GUIDs	Corresponding Folders
1AC14E77-02E7-4E5D-B744-2EB1AE5198B7	C:\Windows\System32
6D809377-6AF0-444B-8957-A3773F02200E	C:\Program Files
7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E	C:\Program Files (x86)
F38BF404-1D43-42F2-9305-67DE0B28FC23	C:\Windows
0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8	C:\ProgramData\Microsoft\Windows \Start Menu\Programs
9E3995AB-1F9C-4F13-B827-48B24B6C7174	%AppData%\Roaming\Microsoft\Internet Explorer\Quick Launch\User Pinned
A77F5D77-2E2B-44C3-A6A2-ABA601054A51	%AppData%\Roaming\Microsoft\Windows \Start Menu\Programs
D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27	C:\Windows\SysWOW64

Table 3: Software tools used in study.

Software tool	Version	Developer	Purpose
VMWare Workstation	11.1.0	VMWare	To create a virtual machine of Windows operating systems
FTK imager	3.4.2.6	AccessData	To extract NTUSER.DAT file from virtual hard disk
Registry Explorer	0.8.1.0	Eric Zimmerman	To parse and view the contents of offline NTUSER.DAT file
UserAssist	2.6.0.0	Didier Stevens	To parse and view contents of UserAssist key of a live system
Registry Editor	5.00	Microsoft	To modify or delete registry subkeys and values
DCode[a]	4.02a	DCode Solutions	To calculate date/time values from binary data
HxD Hex Editor[b]	1.7.7.0	Mael Horz	To view/modify raw content of Amcache.hve file

[a] Available from: http://www.digital-detective.net/digital-forensic-software/free-tools/
[b] Available from: https://mh-nexus.de/en/hxd/

3.3 Tools Used in Study

The software tools used in this research and their purpose are summarized in Table 3.

3.4 Experiments

A number of experiments were formulated to ascertain information required for the objective. These experiments were performed on both type of systems to validate the results highlighted in the subsequent section. Below are the research questions that need to answer through experiments and observations:

- What information is stored when host-based applications are executed?
- What are the effects of installing or uninstalling applications on UserAssist key?
- What information is stored when portable applications are run from various sources such as USB drives and shared network?
- What are the effects of anti-forensic attempts performed on UserAssist key?
- What are the effects of creating a new user account?

3.4.1 Experiment 1 - Running Host-based Applications

This experiment was performed to determine the effects on UserAssist key when host-based applications are run on a Windows system. For this, we executed different host-based applications, such as *Notepad, Wordpad, Run, Microsoft Edge, Windows Media Player* etc. via all possible ways of launching. Note that the applications used are merely examples and can be substituted with countless others. These possible ways of launching a host-based application (Notepad, for our analysis) and its effects on UserAssist key are provided in subsection 4.1.

3.4.2 Experiment 2 - Installing and Uninstalling Applications

For conducting this experiment, we installed different user applications from various sources, such as USB device, Windows store etc., and subsequently launched them to know whether the UserAssist key for these applications behaves differently from host-based applications. To show our analysis, we considered an executable (*MyPublicWiFi.exe*, for example) which was installed with default settings from a USB device and a Windows Store App (*Facebook*, for example) which was installed from Windows Store in Windows 10. Following the installation, these applications were run for multiple times and finally uninstalled from the system.

3.4.3 Experiment 3 - Running Portable Applications

This experiment was conducted to know whether information related to program executions are stored in UserAssist key when portable programs are executed from external devices and shared network. We executed portable applications from a USB device and a shared network. To perform the experiment, the public folder containing various portable applications (in example, *rufus1.4.12.exe*) within user's public profile of a computer with computer-name as *John* was shared over a network. Subsequently, in another computer, these applications were executed from the shared folder.

3.4.4 Experiment 4 - Anti-forensic Attempts

This experiment was designed to observe the behavior of UserAssist key when anti-forensic attempts such as modification and deletion of certain or all UserAssist entries are carried out by a suspected user in order to hide/delete the traces of run applications. Users can modify or delete any entry from UserAssist key using the *Registry Editor* (regedit.exe) application by right-clicking the entry and selecting 'Modify Binary Data' or 'Delete' option. Users can also use *UserAssist* tool to delete particular entries or all. In one experiment, we modify binary data of certain entries including run count, focus count, focus time, and last timestamp of execution. In other experiment, we deleted certain entries from UserAssist key and executed the applications for which the entries were deleted. Before deleting any entry, we noted binary data of that entry to see the changes in binary data after the deletion. One more experiment was conducted in which all UserAssist entries were deleted from the system to know how entries in UserAssist key are populated when applications are executed.

3.4.5 Experiment 5 - Adding a New User Account

This experiment was conducted to investigate the effects of creating a new user account in different versions of Windows system including Windows 7 Professional SP1, Windows 8 Pro and Windows 10 Pro v1607. A standard user account on each considered system was created without signing in with Microsoft account (for Windows 8 and 10). NTUSER.DAT file from each virtual machine was extracted from virtual

hard disk using FTK imager following the system shutdown. For parsing and viewing UserAssist key, *UserAssist* tool was used.

4 RESULTS

This section describes outcomes of the experiments designed in the previous section.

4.1 Running Host-based Applications

Below are the possible ways of launching a host-based application (Notepad, for our analysis) and their effect on UserAssist key.

Action #1: Click Notepad via Start Menu. When Notepad application is run via *Start Menu* then two entries are created; one beneath 'Count' subkey of {CEBFF5CD-ACE2-4F4F-9178-9926F41749EA} and other beneath 'Count' subkey of {F4E57C4B-2036-45F0-A9AB-443BCFE33D9F}. When decoded, first entry is created as *1AC14E77-02E7-4E5D-B744-2EB1AE5198B7\notepad.exe*, while other entry is created as *A77F5D77-2E2B-44C3-A6A2-ABA601054A51\Accessories\Notepad.lnk*. The subsequent runs of the considered application update the binary data of both entries but differently. For instance, focus time in first entry is the total time Notepad applications had focus, in contrast, focus time in other entry was always appeared as zero.

Action #2: Notepad via Shortcut. When the application is run using by doble-clicking *%AppData%\Roaming\Microsoft\Windows \StartMenu\Programs\Accessories \Notepad.lnk* then behavior of UserAssist key was found to be same as for Action #1.

Action #3: Run Directly by Double-clicking on Notepad.exe. When notepad application is run directly by double-clicking on *C:\Windows\System32\notepad.exe*, then information is updated only for *1AC14E77-02E7-4E5D-B744-2EB1AE5198B7\notepad.exe* entry.

Action #4: Notepad >'Run as Administrator' via Start Menu. When notepad application is run as administrator then for entry *1AC14E77-02E7-4E5D-B744-2EB1AE5198B7\notepad.exe* run count increments by three, while for entry *A77F5D77-2E2B-44C3-A6A2-ABA601054A51\Accessories \Notepad.lnk*, run count increments only by one.

Action #5: Notepad via Start Menu >Run... dialog. When notepad application is run via *Start Menu >Run... dialog*, then information is updated only for *1AC14E77-02E7-4E5D-B744-2EB1AE5198B7\notepad.exe* entry.

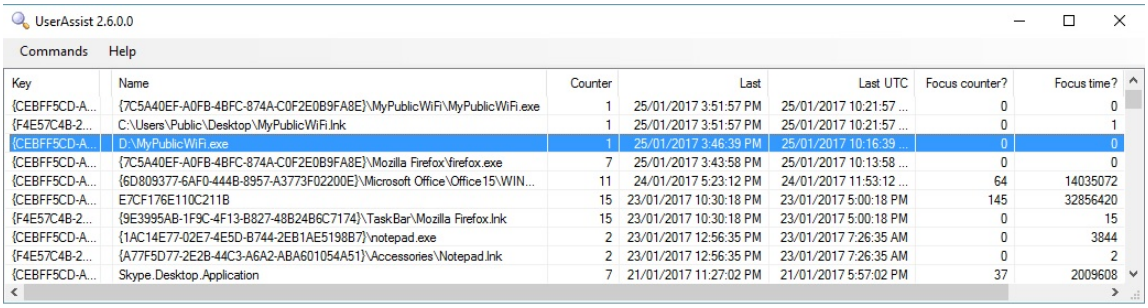
Action #6: Notepad via Taskbar. For this, notepad application is added to Taskbar by clicking *Notepad >'Pin to Taskbar'* and subsequently run from Taskbar. When Notepad is first time launched via Taskbar then run count increments by one for *1AC14E77-02E7-4E5D-B744-2EB1AE5198B7\notepad.exe* entry and new entry 9E3995AB-1F9C-4F13-B827-48B24B6C7174\TaskBar\Notepad.lnk is created with run count as 1, and focus counter as 0 and focus time as 1.

Action #7: Notepad via MFU Start Menu Item. When the application is run using clicking notepad in *Most used list*, then UserAssist key behaves same as for *Start Menu*.

4.2 Installing and Uninstalling Applications

The results of the conducted experiment are as follows:

- If an application is installed on a system, a UserAssist key entry beneath the 'Count' subkey of {*CEBFF5CD-ACE2-4F4F-9178-9926F41749EA*} GUID key is created for that application. The value name for the entry when decoded using ROT-13, represents the complete file path from where the application was installed. The run counter value was found one while focus time and focus counter values were observed as zero. The last execution timestamp represents the date and time the application was installed. Figure 2, illustrates that *MyPublicWiFi.exe* application was installed from *D* drive at timestamp *25/01/2017 3:46:30 PM* (in time zone of local system).
- When any user application, following the installation, is run for the first time then two more UserAssist entries are created for that application. The value name of the entry which is created beneath {*CEBFF5CD-ACE2-4F4F-9178-9926F41749EA*} GUID key signifies file path where the application was installed. While, the entry which is created beneath {*F4E57C4B-2036-45F0-A9AB-443BCFE33D9F*} GUID key signifies the file path to Shortcut (.lnk) file for that application (for our analysis, refer first and second entries in Figure 2). Subsequent runs of an application always update the binary data for key under {*CEBFF5CD-ACE2-4F4F-9178-9926F41749EA*} GUID; however, it may not always update the binary data for key under {*F4E57C4B-2036-45F0-A9AB-443BCFE33D9F*} GUID, due to that fact that the



Key	Name	Counter	Last	Last UTC	Focus counter?	Focus time?
{CEBFF5CD-A...	{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\MyPublicWiFi\MyPublicWiFi.exe	1	25/01/2017 3:51:57 PM	25/01/2017 10:21:57 ...	0	0
{F4E57C4B-2...	C:\Users\Public\Desktop\MyPublicWiFi.lnk	1	25/01/2017 3:51:57 PM	25/01/2017 10:21:57 ...	0	1
{CEBFF5CD-A...	D:\MyPublicWiFi.exe	1	25/01/2017 3:46:39 PM	25/01/2017 10:16:39 ...	0	0
{CEBFF5CD-A...	{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Mozilla Firefox\firefox.exe	7	25/01/2017 3:43:58 PM	25/01/2017 10:13:58 ...	0	0
{CEBFF5CD-A...	{6D809377-6AF0-444B-8957-A3773F02200E}\Microsoft Office\Office15\WIN...	11	24/01/2017 5:23:12 PM	24/01/2017 11:53:12 ...	64	14035072
{CEBFF5CD-A...	E7CF176E110C211B	15	23/01/2017 10:30:18 PM	23/01/2017 5:00:18 PM	145	32856420
{F4E57C4B-2...	{9E3995AB-1F9C-4F13-B827-48B24B6C7174}\TaskBar\Mozilla Firefox.lnk	15	23/01/2017 10:30:18 PM	23/01/2017 5:00:18 PM	0	15
{CEBFF5CD-A...	{1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\notepad.exe	2	23/01/2017 12:56:35 PM	23/01/2017 7:26:35 AM	0	3844
{F4E57C4B-2...	{A77F5D77-2E2B-44C3-A6A2-ABA601054A51}\Accessories\Notepad.lnk	2	23/01/2017 12:56:35 PM	23/01/2017 7:26:35 AM	0	2
{CEBFF5CD-A...	Skype.Desktop.Application	7	21/01/2017 11:27:02 PM	21/01/2017 5:57:02 PM	37	2009608

Figure 2: Parsing UserAssist key using UserAssist tool.

application can be run by other ways instead of double-clicking the .lnk file.

- It was observed that when a user application is launched by the possible ways as listed in Action #1, behavior of the UserAssist entry for that application is reported same as for host-based applications.
- When any user application is uninstalled from the system then the UserAssist entry associated with that application still persists at the same location. If the application was uninstalled by *Control Panel >Programs and Features >Uninstall*, then binary data for that UserAssist entry is not updated with new timestamp. In contrast, if the application was uninstalled by selecting the 'Uninstall' option on the application selected for deletion in Start Menu (or *Start Menu >Uninstall*), then binary data is updated to record the new timestamp. Thus, the UserAssist key may or may not be helpful in determining the uninstall timestamp for the user applications.
- If any Windows store app is installed on a system, no UserAssist entry is created for it even after the system reboot or shutdown. The UserAssist entry for the installed app is created only when the app is run for the first time. Subsequent runs of the app update the binary data for the UserAssist entry as usual.
- When any Windows store app is uninstalled from the system then the UserAssist entry associated with that app still persists at the same location and is not updated with new timestamp. Thus, only last execution timestamp but not the uninstall timestamp could be determined for Windows store apps from UserAssist key.

4.3 Running Portable Applications

Below are the results when the portable applications are run from a shared network, or a USB drive:

- The information related to portable applications is also recorded under {*CEBFF5CD-ACE2-4F4F-9178-9926F41749EA*} GUID of UserAssist key.
- The decoded value names refer the complete file path from which the portable applications were executed including the computer-name of the shared computer over a network (e.g., refer to Figure 3).
- USB drive properties such as drive name and serial number is not recorded in UserAssist key.
- Only the drive letter was observed in the value names of the UserAssist key but not the drive name.
- Other important information such as last execution timestamp and frequency of execution are also recorded that can be helpful in timeline analysis to reconstruct the event.

4.4 Anti-forensic Attempts

Following are the observed effects of anti-forensic attempts performed on UserAssist entries:

- When an entry from UserAssist key is deleted, it was not recreated upon Windows restart or shutdown.
- Entries were recreated only when the applications associated with those entries were executed following the deletion.
- The modifications in binary data of entries persisted until the applications associated with those entries were not executed again.
- The run count, focus count and focus time values were reset when entries were recreated (refer to, Table 4).
- On subsequent runs of applications, binary data of entries were updated as usual.
- When all entries are deleted from UserAssist key, then entries are populated when the applications

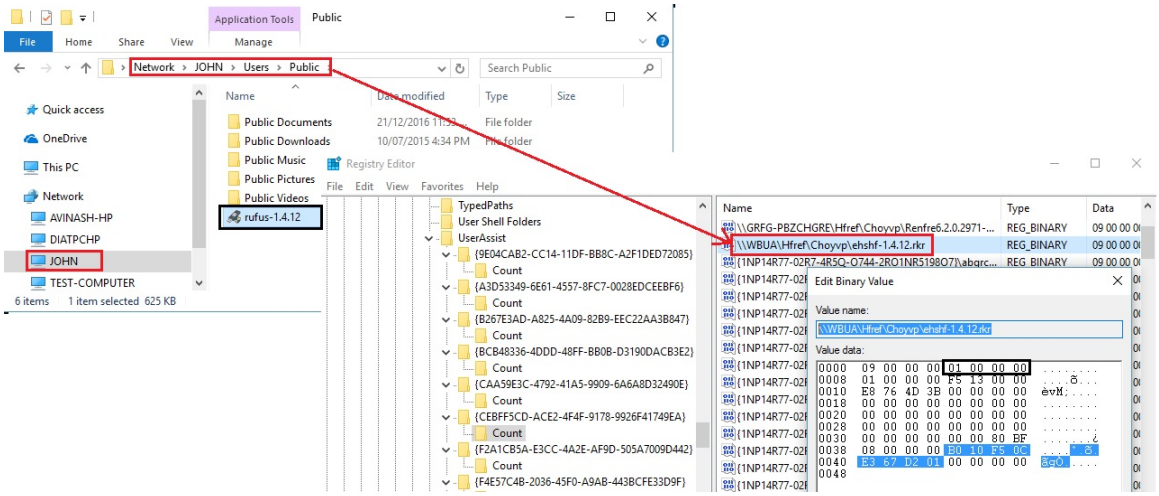


Figure 3: Artifacts of program run from a shared network in UserAssist key.

Table 4: Changes to binary data in UserAssist key entry for WordPad application.

Activities	Timestamp	Run count	Focus time
Run wordpad.exe	25/01/2017 10:43:21 AM	12	67162
Run again	25/01/2017 10:46:11 AM	13	69012
Delete UserAssist entry@11:48, run wordpad.exe@11:49	25/01/2017 10:49:32 AM	1	2926
Run again	25/01/2017 10:51:22 AM	2	4571

are first time executed following the deletions. The information in binary data was reset.

4.5 Adding a New User Account

The observed effects of adding a new user account are as follows:

- Setting up a new user account in a Windows system results in creation of NTUSER.DAT hive file in their profile. The location of UserAssist key was observed same as for original users.
- The contents of new UserAssist key was found different from the UserAssist key for the original user.
- The last execution timestamp in entries of default applications was observed same which is the timestamp of setting up the new user account.
- The creation of new user account did not affect the content of UserAssist key for the original user.

5 DISCUSSION

The research into the forensic usefulness of various resources related to program execution analysis in Windows has been conducted actively in digital forensics community. This section presents and compares the forensic capabilities of such resources investigated in published research articles and blog posts. Six sources of program executions namely, IconCache.db (Collie, 2013) (Lee and Lee, 2014), Shortcut (Singh and Singh, 2016a) (Parsonage, 2008), Prefetch (Wade, 2010), SRUDB.dat (Khatri, 2015), Amcache.hve (Singh and Singh, 2016b) and UserAssist key are taken into consideration..

The IconCache.db files store information of file paths of applications that have been installed, run, copied or viewed by a user on Windows systems. The information related to timestamp of installed or executed applications is not recorded in IconCache.db files. Shortcut (.lnk) files record information related to applications including created, accessed and modified timestamps (in UTC), full file path, drive type, serial number, and the volume name from which the applications were run. However, Shortcut (.lnk) files do not store any information related to run count of applications. A Prefetch file can provide information of file path from which the application was executed, timestamps of the first and the last execution, and the number of times certain application has been executed on system (Wade, 2010). However, Prefetch files are not always created for Window store apps as many apps are not associated to an executable file. Also, if

Table 5: Comparison of artifacts stored in IconCache.db, Prefetch, Shortcut, SRUDB.dat, Amcache.hve and UserAssist.

	Full execution file path	Installed timestamp	Last run timestamp	Run count	Uninstall timestamp	Traces of deleted applications	Traces of Windows Store App runs
IconCache.db	✓	×	×	×	×	⋆	×
Prefetch	✓	×	✓	✓	×	✓	⋆
Shortcut (.lnk)	✓	✓	✓	×	×	×	×
SRUDB.dat	✓	×	✓	✓	×	×	✓
Amcache.hve	✓	✓	✓	×	✓	✓	×
UserAssist	✓	⋆	✓	✓	⋆	✓	✓

[a] ✓– can be determined, ×– cannot be determined, ⋆– may or may not be determined

the applications are deleted from the system, Prefetch files do not record timestamp of deletions of applications. It is important to note that UserAssist key run count differs from Prefetch run count, as UserAssist key is user-specific. The SRUDB.dat file in Windows 8 was first investigated in (Khatri, 2015), as a forensic resource that can be useful in tracking in tracking external or deleted applications, Windows store app runs and user process mapping. However, the information in this database lasts only for one month. By combining information from SRUDB.dat and Prefetch file for a certain application, forensic analysts can estimate how long the process was running and know the user who launched it. The Amcache.hve is a registry hive file that can reveal information such as install timestamp of applications, full execution path, timestamp of the last time certain application was run, traces of deleted applications, including the timestamp of deletions. However, the run count of applications is not recorded in this artifact. Fortunately, the UserAssist key records more information of program executions than any other source considered. For example, UserAssist key can provide information of an application's run count and the traces of deleted applications including the timestamp of deletions for portable executable files, package applications and Windows store apps, which cannot be gathered from any other considered source. Nevertheless, the UserAssist key cannot always reveal the timestamps when the certain application was installed or uninstalled from the Windows system. Thus, forensic analysts are suggested to combine and correlate the information from abovementioned sources for better tracking of program executions.

6 CONCLUSIONS

Program execution analysis often proves to be a meaningful effort that unravels the story of what happened on a system, including the furtive behaviors meant to complicate the investigations. The UserAssist key records information related to programs executed on a Windows system that can be helpful while investigating cases related to digital forensics and incident response.

The number and type of UserAssist key entries can reveal a lot more about a user who is using Windows system. For instance, if the UserAssist key entries are relatively less (about ten to fifteen) then investigator can infer that the system is relatively new and only a few applications have been executed. Typically, this condition may be of a normal Windows user. In contrast, if the UserAssist key entries are relatively more than of a normal user then analysts can infer that the system has been extensively used and numerous different applications have been executed over a short or long period of time. This situation is indicative that the user is a technical user. Thus, the number and type of UserAssist key entries can help in identifying users- profiling the user's technical capabilities. For example, the existence of programming applications such as *Perl*, *Ruby* or *Python* would be indicative that the user is highly technical while the existence of vulnerability analysis programs such as *Metasploit*, *netcat*, *nmap* can reveal the nature and possibly, the intention of the Windows user.

The paper presents the binary structure of UserAssist key in modern Windows and compares it with that in older version of Windows (e.g., Windows XP). A number of experiments were conducted to investigate the behavior of UserAssist key when applications are executed from different sources, such as external device, Windows store, and shared network. The behavior of UserAssist key will have direct implications on forensic investigations involving program execution analysis as a subject of interest. The UserAssist key records a great deal of information that can be extremely useful evidence in digital investigations, where program execution analysis is of investigators' prime interest. The information from UserAssist key, such as run count and focus time can provide clue on the frequency and the total time an evil program was executed on the system in question. We found that evidence of program execution persisted in UserAssist key even after the target applications have been removed from the system. Further, we discussed that UserAssist key can be a useful resource for profiling

the user's technical capability, based on the number and the type of applications installed on the system. The paper also provides the comparison of forensic capabilities of various sources of program execution analysis, namely, IconCache.db, Prefetch, Shortcut (.lnk), SRUDB.dat, Amcache.hve, and UserAssist. This is intended to help analysts to combine and correlate the information from these sources for better tracking of applications. However, analysts are suggested to corroborate the findings from other sources, such as *Jump Lists*, *AppCompatFlag*s, *AppCompatCache*, *Event Logs*, *Pagefile.sys*, and *USN Journal*.

REFERENCES

AccessData (2008). Understanding the userassist registry key. https://ad-pdf.s3.amazonaws.com/UserAssist%20Registry%20Key%209-8-08.pdf. [accessed 12-Jan-2017].

Carvey, H. (2005). The windows registry as a forensic resource. *Digital Investigation*, 2(3):201–205.

Carvey, H. (2011). *Windows registry forensics: Advanced digital forensic analysis of the windows registry*. Elsevier.

Carvey, H. (2013). Regripper. https://code.google.com/archive/p/regripper/downloads. [accessed 26-Jan-2017].

Carvey, H. (2016). *Windows registry forensics: Advanced digital forensic analysis of the windows registry*. Elsevier, 2nd edition.

Carvey, H. and Altheide, C. (2005). Tracking usb storage: Analysis of windows artifacts generated by usb storage devices. *Digital Investigation*, 2(2):94–100.

Collie, J. (2013). The windows iconcache.db: A resource for forensic artifacts from usb connectable devices. *Digital Investigation*, 9(3):200–210.

Eterovic-Soric, B., Choo, K.-K. R., Mubarak, S., and Ashman, H. (2017). Windows 7 antiforensics: A review and a novel approach. *Journal of Forensic Sciences*.

Khatri, Y. (2015). Forensic implications of system resource usage monitor (srum) data in windows 8. *Digital Investigation*, 12:53–65.

Lee, C.-Y. and Lee, S. (2014). Structure and application of iconcache.db files for digital forensics. *Digital Investigation*, 11(2):102–110.

Lees, C. (2013). Determining removal of forensic artefacts using the usn change journal. *Digital Investigation*, 10(4):300–310.

Mee, V. and Jones, A. (2005). The windows operating system registry–a central repository of evidence. In *Proceedings from e-crime and computer evidence conference*, volume 2005.

Mee, V., Tryfonas, T., and Sutherland, I. (2006). The windows registry as a forensic artefact: Illustrating evidence collection for internet usage. *digital investigation*, 3(3):166–173.

Parsonage, H. (2008). The meaning of link files in forensic examinations. *Computer Forensic misCellany*.

Pullega, D. (2013). Userassist forensics (timelines, interpretation, testing, & more). http://www.4n6k.com/2013/05/userassist-forensics-timelines.html. [accessed 29-Dec-2016].

Singh, B. and Singh, U. (2016a). A forensic insight into windows 10 jump lists. *Digital Investigation*, 17:1–13.

Singh, B. and Singh, U. (2016b). Leveraging the windows amcache.hve file in forensic investigations. *The Journal of Digital Forensics, Security and Law: JDFSL*, 11(4).

Stevens, D. (2006). Userassist. https://blog.didierstevens.com/programs/userassist/. [accessed 02-Dec-2016].

Stevens, D. (2010). Windows 7 userassist registry keys. https://intotheboxes.files.wordpress.com/2010/04/intotheboxes_2010_q1.pdf. [accessed 02-Feb-2017].

Talebi, J., Dehghantanha, A., and Mahmoud, R. (2015). Introducing and analysis of the windows 8 event log for forensic purposes. In *Computational Forensics*, pages 145–162. Springer.

Wade, M. A. (2010). Decoding prefetch files for forensic purposes. *Harris Corporation*.

Wong, L. W. (2007). Forensic analysis of the windows registry. *Forensic Focus*, 1.

Zimmerman, E. (2015). Registry explorer/recmd version 0.7.1.0. https://ericzimmerman.github.io/. [accessed 26-Jan-2017].