Investigating file use and knowledge with Windows 10 artifacts

A. Đuranec[1], D. Topolčić[1], K. Hausknecht[1], D. Delija[2]
[1] INsig2 d.o.o., Zagreb, Croatia
[2] Zagreb University of Applied Sciences, Zagreb, Croatia
Antun.Duranec@insig2.com, Davorka.Topolcic@insig2.com, Kresimir.Hausknecht@insig2.com, Damir.Delija@tvz.hr

***Abstract* - Windows 10 operating system is the most widely used operating system today that contains many programs and mechanisms for managing computer hardware and software. Looking from a digital forensics point of view these produce valuable records of user activities. In a forensic world, such records are known as Windows artifact which can be described as a system generated records of the user activities that have forensic value. Gaining a deep understanding of how these records are created and what information they contain can help the examiner to acquire valuable data that can be used as evidence and support other evidence. The artifacts can be a great way to focus on relevant data and reduce the need for full examination of constantly increasing data storage that examiners encounter. Through this paper, the focus will be on analyzing different, fewer know artifacts, that aren't supported by mainstream forensic tools because they vary between versions of Windows, resulting in the need for manual analysis. Their deep understanding is necessary to avoid misinterpreting their content which can result in wrong conclusions. Additionally, the paper presents the results of testing Windows 10 artifacts and open-source tools used in the testing process.**

Keywords - Windows 10; artifacts; digital forensics

I. INTRODUCTION

Digital forensic examiners in Windows operating system investigations use forensic tools that parse Windows artifacts automatically. These artifacts often differ between versions of Windows system. Sometimes forensic tools won't be able to parse this valuable information automatically. Through this paper, the value of manually analyzing Windows artifacts will be shown, as well as how they can be used to separate potentially relevant data for investigation. The focus will be on useful information, for file use and knowledge, that can be found in specific artifacts and how their deep understanding is needed to avoid misinterpreting the findings.

The paper is organized as follows. Section 2 and 3 give a brief overview of digital forensic and Windows 10 operating system. Section 4 describes specific Windows artifacts and their value in forensic investigations, while section 5 combines before explained artifacts in investigation of file use and knowledge use-case. Section 6 concludes the paper.

II. DIGITAL FORENSICS

Digital forensic, also known as digital forensics science is a branch of forensic science encompassing the recovery and investigation of material found in digital devices, often in relation to computer crime. The history of digital forensics goes back to late 1980s and early 1990s as a result of the adaptation of personal computers by a wide range of people, industry, and appearance of digital crime. The growth of digital crime caused law enforcement agencies to begin establishing specialized groups to handle technical aspects of investigations [1].

Because of the fast and rapid spread of electronic devices, today it's hard to imagine a crime that doesn't involve digital evidence. Criminals are using technology to facilitate their offenses and avoid apprehension, creating new challenges for law enforcement agents and forensics examiners [2]. As technology and electronic devices advanced through years, so did digital forensic which can today be separated in few core branches; computer forensics, mobile forensics, network forensics, and cloud forensics, that are related to the type of digital devices involved in digital crime. The digital forensic process starts after a crime is committed and noticed; typically process includes preparation, identification of potential evidence, seizure and acquisition, analysis and report or presentation of findings.

III. WINDOWS 10 OPERATING SYSTEM

An operating system is a collection of software that manages computer resources and provides services for computer programs.

From 1985 till 2019 Microsoft developed 26 versions of the Windows operating system. Each version of the operating system brought changes to the user interface and equally, most of them had a different way of storing data that is of forensic value to the examiner. Microsoft stopped mainstream support for all Windows systems except for Windows Server 2016, 2019 and latest, Windows 10 [3]. Windows 10 is a series of personal computer operating systems which was released in July of 2015; Windows 10 receives new builds on an ongoing basis at no additional cost to users. Now, the latest build of this operating system is 1809; labeled YYMM where YY represents the year and MM represents the month of release [4]. Windows 10 introduced many new features like the timeline, focus assist, nearby sharing, new parental control, touch input, face login and many more

TABLE 1. WINDOWS OPERATING SYSTEM SHARE BY VERSION

Month	Windows 7	Windows 10
June 2018	**17.98%**	15.38%
July 2018	**18.66%**	16.56%
August 2018	**16.46%**	15.44%
September 2018	**17.53%**	16.06%
October 2018	**16.63%**	15.92%
November 2018	**16.08%**	15.77%
December 2018	15.09%	**16.04%**

[5]. As shown in the Table 1, in December 2018 Windows 10 overthrow Windows 7 as most widely used Windows system which means that digital forensic examiners will most likely encounter mentioned operating system; however, even after the end of official support by Microsoft, examiners will likely encounter older versions which require understanding of differences between system version.

IV. WINDOWS 10 ARTIFACTS

From a digital forensic examiner view, Windows artifacts are evidential records that are automatically generated and saved by the Windows operating system as a result of user interaction with the computer. Each version of Windows operating system contains many different artifacts that differ between each version. Some of the artifacts are found in all versions while some versions on first look don't support specific artifacts, although they can still be present because of the need for backward compatibility or future use. Often a problem for mainstream forensics tools are changes in data format and location of artifacts that happen between versions, demanding the need for vendors to often update their tools which in some cases is not acceptable and need for manual analysis is required.

From the large list of Windows artifacts available in Windows 10 operating system, the focus will be on manual analysis of Shimcache, Prefetch, Shellbag, $UsnJrnl, System Resource Usage Monitor (SRUM) and UserAssist artifacts with open-source tools.

A. Shimcache

This cache (also known as AppCompatCache) is a component of the Application Compatibility Database which is used to identify application compatibility issues and track executables on Windows system. This artifact contains the name and file path of executables on the system. Poor understanding of this artifact can be misleading as records of executables don't confirm that they were actually executed, rather, it means that specific executables were recognized by the system. In records, the examiner can find associated "last modified" date and time value; however, this value represents modified time of the file; e.g. for NTFS, this is the last modified time of the file as stored in the $STANDARD_INFORMATION attribute. Furthermore, new entries are recorded at the moment of system shut down with the most recent entries placed on top of the list. This artifact records up to 1024 entries and only files with specific extensions are logged (e.g. ".bat", ".exe", ".dll"). Passing maximum 1024 entries will result in the old entries being overwritten by the new ones. The location of above-mentioned information is recorded in Windows registry [1] under registry key "HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatCache\AppCompatCache". The file that contains mentioned registry key is "SYSTEM" registry file located at "C:\Windows\System32\config" folder.

There are many tools for parsing Shimcache data, commercial and open-source, and analysis for the purpose of this paper has been done with Eric Zimmerman's AppCompatCacheParser (v1.4) tool [6]. When the tool is run through command prompt and required attributes are specified, it will parse "SYSTEM" hive's registry key AppCompatCache and all records will be exported in comma-separated values (CSV) file. The CSV file will contain the cache entry position for each entry, full path to the file and last modified time (modification time applies to that of the file) in Coordinated Universal Time (UTC) as shown in Table 2. Additionally, executed flag in Windows 10 (1809) is not present and they all have the value of "NA".

Microsoft changed the data format of Shimcache through versions of Windows system resulting in the need for detailed testing and adjustment of forensic tools. Reference [7] is Application Compatibility Cache Key's knowledge base which contains known information about this artifact and how it has been changed through different versions of Windows system. This artifact is great for determining executables and malicious behavior that was present on the system, even if they were deleted. Furthermore, it also shows directories where executables were located which can help examiner detect purposely buried folders.

B. Prefetch

Prefetch feature has been introduced with Windows XP operating system and it's still present in Windows 10; it enables Windows to load portions of frequently used programs when the computer first boots up, enabling them to load faster.

Windows prefetch, in the first 10 seconds of program

TABLE 1. SHIMCACHE ENTRIESPARSED APP COMPAT CACHE RECORDS

Cache Entry Position	Path to the file	Last Modified Time (UTC)	Executed
0	.\VeraCrypt.exe	17.1.2019 10:34:58	NA
1	.\FTK Imager.exe	20.10.2017 15:08:38	NA
2	.\firefox.exe	17.1.2019 9:50:14	NA
3	.\ZoomIt.exe	18.6.2013 13:12:34	NA
4	\Bandizip.exe	2.10.2018 11:11:49	NA
5	iexplore.exe	15.9.2018 17:39:41	NA

[1] The Windows Registry is a hierarchical database that stores low-level settings for the Microsoft Windows operating system.

execution, records name of application, date and time of executions, run count and path to executable file. Time and date information is collected on the first run of a specific application and on every next run the "last run time" is updated. Extension of prefetch file is ".pf" and the name format of the file is "<*AppName*>.EXE-XXXXXXXX.pf" where "XXXXXXXX" represents the hash value of executable's location path. The system can contain multiple prefetch files of an identical program because of different locations of executables, resulting in prefetch file with same application name but different hash value. Forensics examiners can encounter system without prefetch artifacts for two reasons; user manually turned off prefetching in Windows Registry or sometimes system has prefetching disabled because of the Solid-State Drive (SSD). Prefetch files are located at "C:\Windows\Prefetch" and their parameters are set up in Windows Registry key "HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\MemoryManagement\PrefetchParameters".
Windows 10 introduced changes of prefetch format and in which way the file is stored on disk; it uses compression to store prefetch files, and signature is "MAM." or in hexadecimal "4D 41 4D 04". The most important change is that prefetch now stores up to eight "Last run" times.

Testing has been completed with Eric's PECmd (v1.2.0.2) tool which parses prefetch artifacts [8]. As result this command prompt (CMD) tool export a CSV file that contains the full path of processed prefetch file, its created, modified and accessed time, executable name and path's hash, version of operating system, run count, run dates and times for up to last eight executions, as shown in Table 3. Examiner can use prefetch to determine how many times the program was run from a specific location, even if the program is deleted. Additionally, it can contain evidence of wiping tools and in case of malware execution examiner can determine first/last run times and location of malware executable.

C. *Shellbag*

Microsoft Windows uses a set of Registry keys known as "Shellbags" to maintain the size, views, icon, and position of a folder when using Windows Explorer [9]. It tracks and stores how different windows appear to the user when they view it; to give constant experience between reboots of the system. From forensic examiner point of view, it's a great artifact for quick profiling of the system and determining which folders did user access. It records information as folder position, size, icons, associated date and time in a moment of change. Shellbag data is user-driven and it also tracks local network and external directories. Data will remain within Registry key even after a folder is deleted or Universal Serial Bus (USB) flash drive is detached, and it's user account specific. Shellbag data is recorded in NTUSER.DAT and UsrClass.dat files; NTUSER.DAT file is located at "C:\Users\<*User*>" and UsrClass.dat at "C:\Users\<*User*>\AppData\Local\Microsoft\Windows".

TABLE 2. PARSED PREFETCH DATA

C:\Windows\Prefetch\HXD.EXE-3AAB4DBE.pf	
Source Created (UTC)	24.1.2019 11:06:52
Run Date & Time (UTC)	24.1.2019 12:08:28 24.1.2019 12:08:24 24.1.2019 12:03:59 24.1.2019 11:35:28 24.1.2019 11:24:23
Run Count	5
Version	Windows 10
Hash	3AAB4DBE

TABLE 3. SHELLBAGS BEHAVIOR

Entry Creation (Desktop)	
Activity	Shellbags recorded
Creation of a folder	NO
Right click on a folder	YES
Opening a folder	YES
Copy & Cut function	YES
Rename (double click on name)	NO
Rename (right click - rename)	YES
Deletion of a folder	YES
Selecting a folder	NO
Additional Entry Creation (folder structure)	
Activity	Shellbags recorded
Creation of a folder	YES
Selecting a folder	YES
Renaming of a folder	YES
Entry Creation (CMD)	
Activity	Shellbags recorded
Creating a folder	NO
Deletion of a folder	NO
Renaming of a folder	NO

On Windows 10 (1809) system, after loading the NTUSER.DAT file in NirSoft's ShellBagsView (v1.21) tool [10], no Shellbags records were present; however, loading the UsrClass.dat file showed all Shellbags records. Shellbags data is stored and updated on various user's activities, and it also depends on whether the user is manipulating folders directly on desktop or folder structure. Table 4 shows when Shellbags data is recorded in a moment of different user's activities.

D. *Update Sequence Number Journal ($UsnJrnl)*

UsnJrnl is a feature of the New Technology File System (NTFS) which maintains the record of changes made to the volume [11]. When enabled, the system records all changes made to files and directories on volume in the UsnJrnl; creation, deletion, modification, overwrite, compression, etc. This artifact is a great source of timeline information for malware and incidence response investigations, time stomping and anti-forensics activities. From Windows Vista and onwards the UsnJrnl is enabled by default. It's composed of "$Max" and "$J" data streams located in hidden system file at "C:\$Extend\$UsnJrnl". Data stream "$Max" contains metadata of change log while “$J” contains actual change log records including Update Sequence Number (USN) and Master File Table (MFT) reference number [12]. The USN information is also stored in $STANDARD_INFORMATION attribute of the MFT record. Additionally, the MFT header contains $Logfile Sequence Number (LSN) [13]. The Blueangel’s NTFS Log Tracker uses a combination of $J, $LogFile and $MFT files to display more valuable information [14]. NTFS Log Tracker extract UsnJrnl timestamp, USN, file name, event flags, source info, and file attribute. It also records the full path of the file, extracted from MFT, if selected. List of possible event flags, file attributes and source info can be found in official Microsoft documentation [15], [16].

E. System Resource Usage Monitor (SRUM)

SRUM (also known as SRUDB.dat) feature has been presented in Windows 8 operating system that tracks application resource usage, services, windows apps, energy usage, networks connections, and data usage by maintaining the database of historical activity. The file is an Extensible Storage Engine (ESE) database that has multiple database tables as shown in Table 5 [17]. This artifact records names and paths of every application that executes on the system. Its records user’s Security Identifier[2] (SID) that executed the program. Furthermore, it records the names of all networks on which system has been connected, length of connection and how many bites were written to or read from the hard drive by the specific application. Its records application battery usage and Central Processing Unit (CPU) usage [18]. This artifact can help the examiner to determine which user launched a specific process, data upload and download, it keeps data of programs that were deleted, installed, uninstalled, run from external media, etc. The entries are written to the file (date and time) every hour and on shutdown, meaning that examiner won’t get the exact time when the application has been run but when operating system recorded it. The data is temporarily saved in Windows Registry at Registry key ”/HKLM\SOFTWARE\Microsoft\WindowsNT\-CurrentVersion\SRUM\Extension” and then it’s periodically transferred in the SRUDB.dat database at “C:\Windows\System32\sru\SRUDB.dat. For the purpose of this paper, analysis of the SRUM data has been done with Nirsoft’s ESEDatabaseView (v1.62) utility that reads

TABLE 4. IMPORTANT SRUMDB.DAT TABLES

Table GUID	SRUM Extension
{973F5D5C-1D90-4944-BE8E-24B94231A174}	Network Data Usage Monitor
{D10CA2FE-6FCF-4F6D-848E-B2E99266FA89}	Application Resource Usage Provider
{DD6636C4-8929-4683-974E-22C046A43763}	Network Connectivity Usage Monitor
{D10CA2FE-6FCF-4F6D-848E-B2E99266FA86}	Push Notifications (WPN) Provider

and displays the data stored in the ESE database [19]. Opening an SRUDB.dat file in the above-mentioned utility will show all present tables with their Globally Unique Identifier (GUID). To get valuable data from the ESE database, the forensic examiner needs to have a good understanding of how SRUM tables are connected among each other. For example, checking the Application Resource Usage table won’t contain application name and user SID, but it will contain “AppID” and “UserID” columns. These IDs can also be found in “SruDbIdMapTable” table under the column “IdIndex” and the corresponding column “IdBlob” contains actual data like application name with path and user SIDs, depending on the value of “IdType”, in format of 16-bit Unicode Transformation Format (UTF-16) encoded string [16]. ESEDatabaseView utility has the functionality to detect UTF-16 strings in binary data that will convert “IdBlob” content in the readable format. However, the mentioned functionality won’t show user’s SID in the readable format. SIDs are identified with “IdType” value 3 and they need to be manually converted to usual string format as they are composed of multiple components as shown in Table 6 [20]. Converting SIDs components can be done with Microsoft Calculator in programmer mode. Once when the examiner has the user’s SID and application name, the timeline of specific user activities can be determined.

F. UserAssist

UserAssist is a Registry key that records the use of applications by the user. Based on the records of this Registry key, Windows Explorer displays frequently used programs on the left side of the Start menu in the Windows XP system. It’s quite similar to the before mentioned Prefetch artifact with the biggest difference that it’s user-specific, unlike Prefetch. To be added in UserAssist, applications need to be launched over a shortcut, link file, explorer, etc. Programs run through CMD won’t be recorded. UserAssist is located in NTUSER.DAT file at “C:\Users\<users>” folder and the specific Registry key that store UserAssist data is located at “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist”. The UserAssist

TABLE 5. RAW HEX SID COMPONENTS

Raw Hexadecimal SID example	
0105000000000005150000007C8906DF25E4B91C52E245D7E9030000	
SID’s component	Format and Value
SID identifier	**S**
Version of SID	0x01 = **1**
Number of dashes (minus 2)	0x00 00 00 00 00 05 (48-bit number in big-endian format) = **5**
Identifier authority	0x15 00 00 00 (32-bit number in little-endian format) = **21**
Domain or local computer identifier	0x7C 89 06 DF - 25 E4 B9 1C - 52 E2 45 D7 (32-bit in little-endian) = **3741747580-481944613-3611681362**
Relative ID (RID)	0xE9 03 00 00 (32-bit in little-endian) = **1001**
Converted SID	S-1-5-21-3741747580-481944613-3611681362-1001

[2] Security Identifier is a number used to identify user, group, and computer accounts in Windows.

Registry key contains multiple sub-keys represented with GUIDs as shown in Table 7 [21], [22]. The values of this Registry key are ROT 13 ciphered, meaning that examiner will need to use tools that can decipher them [23]. The UserAssist contains the program's path, run counter, last run time, focus counter and focus time. This artifact is great for determining the frequency of program execution, the time of the last program launch, deletion of a program, how long a user has interacted with a given program and many more user behaviors. It's a great source of information on servers and computers which have Prefetch disabled. Testing of this artifact has been done with Didier Stevens's UserAssist (v2.6.0) tool that can decipher and parse this artifact on a live system or by importing NTUSER.DAT file [24]. To successfully open *.DAT file, the tool must be run with administrator privileges and sometimes the loading of *.DAT file can fail; the tool has the option to load *.reg files which have been loaded successfully on each try. The UserAssist utility displays Registry key, the index number of entries, run counter, system last run time and last run time in UTC, focus counter and focus time. Executed program's path can sometimes include folder GUIDs; a few examples are shown in Table 7.

From testing, it has been concluded that UserAssist data is recorded immediately and that focus counter represents how many times specific run program's window was selected by the user while focus time represents total time, in milliseconds, that user had the window selected. The UserAssist as many artifacts have few exceptions that require good understanding. For an example, UserAssist won't increase run count if the program is run on startup or as a scheduled task; however, the entry will be recorded with focus counter and time. Furthermore, it's possible, because of above-mentioned exceptions that run counter will have a value of "0".

V. Combining the Artifacts

Examiner should always try to combine multiple artifacts; their combining will result in a better understanding of user activity and they will mutually support each other.

An example below is showing how artifacts can be combined and used to investigate use file and knowledge. For this purpose, a small use case has been created where it's suspected that a user was in possession of confidential document; however, files and wiping tools were not found on suspects system on initial check. To determine if that is the case, a combination of the before mentioned artifacts have been used.

With checking Shimcache data examiner can determine if there has been the presence of non-standard executables on the system. By looking at extracted Shimcache records it can be determined that the wiping tool "Eraser" installation file was present as well as a record of the executable file of the mentioned wiping tool as shown in Table 8. With Shimcache records examiner can confirm that user had executable files of wiping tool present on the system and there is a possibility that the program has been run. To determine if wiping program has been run examiner can look in Prefetch, UserAssist, SRUM and UsnJrnl records. In the Prefetch folder, two Eraser Prefetch files have been created by the operating system. As Prefetch files are created on the first run of a program, the examiner can confirm that the wiping tool has been run and when. The created time of file run counter and last run times gives examiner more insights on usage of the wiping tool as shown in Table 9. To support these findings examiner can as mentioned before, look at SRUM records. The first step is determining "IdIndex" of Eraser wiping tool in table "SruDbIdMapTable" of SRUM database, and then the matching "IdIndex" number must be found in application resource usage table under "AppId". Corresponding "UserID" can be found in column right of it, and then examiner can identify users SID in table "SruDbIdMapTable". When above mentioned SRUM data is connected examiner can once again confirm that specific wiping tool has been run and by which SID; as SID number is unique for each user, the user that run program can be identified as shown in Table 10.

The third artifact that can confirm that the wiping tool was run is the UserAssist. With it, examiner can find records of launched programs by a specific user. Additionally, the examiner can also determine last run time, path to executable file, run count and how long the

TABLE 7. USERASSIST SUB-KEY AND KNOWN FOLDER GUIDS

Sub-keys GUIDs	Description
{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}	Executable file execution
{F4E57C4B-2036-45F0-A9AB-443BCFE33D9F}	Shortcut file execution
Known folder's GUIDs	Description
{0139D44E-6AFE-49F2-8690-3DAFCAE6FFB8}	Common programs
{1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}	System
{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}	Program files x86
{6D809377-6AF0-444B-8957-A3773F02200E}	Program files x64
{9E3995AB-1F9C-4F13-B827-48B24B6C7174}	User pinned
{A77F5D77-2E2B-44C3-A6A2-ABA601054A51}	Programs
{F38BF404-1D43-42F2-9305-67DE0B28FC23}	Windows

TABLE 8. SHIMCACHE RECORDS

Entry Position	Path	Modified Time UTC
2	C:\Program Files\Eraser\Eraser.exe	28.8.2016 0:50:44
5	C:\Users\Admin\Desktop\Eraser 6.2.0.2979.exe	13.4.2018 13:09:43

TABLE 9. PREFETCH RECORDS

File Name	Created Time	Run Counter	Last Run Time
ERASER 6.2.0.2979.EXE-D37D8F97.pf	5.2.2019. 11:56:42	2	5.2.2019. 12:02:14, 5.2.2019. 11:56:35
ERASER.EXE-CE61944A.pf	5.2.2019. 11:57:44	1	5.2.2019. 11:57:34

TABLE 10. SRUM RECORDS

SruDbIdMapTable		
IdType	IdIndex	IdBlob
0	1098	.\Eraser.exe
0	1154	.\Eraser 6.2.0.2979.exe
3	334	010500000000000515000000 7C8906DF25E4B91C52E245 D7E9030000
Application Resource Usage table		
TimeStamp	AppId	UserId
05.02.2019 12:04:00	1098	334
05.02.2019 12:04:00	1154	334

TABLE 11. USERASSIST RECORDS

Name	Counter	Last	Focus Counter
.\Eraser 6.2.0.2979.exe	2	5.2.2019 12:02:14	11735
.\Eraser.exe	1	5.2.2019 11:57:33	251891

wiping tool was in focus of screen; as shown in Table 11.

UsnJrnl tracks all changes that are made to files and directories on the volume which an examiner can use to determine the activity on the system. In this case, UsnJrnl contains a record of confidential file's link creation and WordPad program prefetch file being changed. Record also shows that confidential file has been renamed and when Eraser files and its Prefetch file were created. Furthermore, there are records of constant file renaming with random characters, ending with deletion of the confidential file which confirms wiping of confidential data.

With all collected information examiner can confirm the presence of confidential data and wiping program for hiding evidence.

VI. CONCLUSION

Mentioned Windows artifacts have a great value in forensic investigations, especially when it comes to understanding file use and knowledge; it's crucial for an examiner to understand how, when and for what purpose specific files and programs have been used. These systems generated records offer examiner a way to do quick profiling of user activities and reduce the time of analysis, but their deep understanding is required. Furthermore, the artifacts often change between versions of the Windows operating system and manual analysis of the artifacts sometimes can't be avoided.

To acquire a better understanding of Windows artifacts examiners should always analyze changes, test artifacts behavior, and their forensic tools so that they can avoid misinterpreting valuable data.

REFERENCES

[1] Digital forensics, (2019, January). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Digital_forensics

[2] E. Casey, Digital Evidence and Computer Crime, 3rd ed., Elsevier, 2011, pp. 3.

[3] List of Microsoft Windows versions, (2019, January), Retrieved from Wikipedia: https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_versions

[4] Windows 10, (2019, January), Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Windows_10

[5] M. Muchmore, Microsoft Windows 10, 2019, Retrieved from PCMag: https://www.pcmag.com/article2/0,2817,2488631,00.asp

[6] E. Zimmerman, (2019, January), Eric Zimmerman's tools, https://ericzimmerman.github.io/#!index.md

[7] J. Metz, (2017, April), Application Compatibility Cache Key, https://github.com/libyal/winreg-kb/blob/master/documentation/Application%20Compatibility%20Cache%20key.asciidoc

[8] E. Zimmerman, (2016, January), PECmd v0.6.0.0 released, https://binaryforay.blogspot.com/2016/01/pecmd-v0600-released.html

[9] W. Ballenthin (2019, January), Windows Shellbag Forensics, http://www.williballenthin.com/forensics/shellbags/

[10] NirSoft, (2019, January), ShellBagsView v1.21 https://www.nirsoft.net/utils/shell_bags_view.html

[11] USN Journal, (2019, January), Retrieved from Wikipedia: https://en.wikipedia.org/wiki/USN_Journal

[12] J. Oh, (2019, January), Advanced $UsnJrnl Forensics, http://forensicinsight.org/wp-content/uploads/2013/07/F-INSIGHT-Advanced-UsnJrnl-Forensics-English.pdf

[13] BlackBag Training Team, (2017, May) Master File Table Basics, https://www.blackbagtech.com/blog/2017/05/02/master-file-table-basics/

[14] J. Oh, (2019, January), blueangel's ForensicNote, https://sites.google.com/site/forensicnote/ntfs-log-tracker

[15] USN_RECORD_V4 structure, (2018, December), https://docs.microsoft.com/en-us/windows/desktop/api/winioctl/ns-winioctl-usn_record_v4

[16] File Attribute Constant (2018, May), https://docs.microsoft.com/en-us/windows/desktop/FileIO/file-attribute-constants

[17] J. Metz, (2019, January), System Resource Usage Monitor (SRUM) database, https://github.com/libyal/esedb-kb/blob/master/documentation/System%20Resource%20Usage%20Monitor%20(SRUM).asciidoc

[18] M. Baggett, (2019, January), System Resource Utilization Monitor, https://isc.sans.edu/forums/diary/System+Resource+Utilization+Monitor/21927

[19] NirSoft. (2019, January), ESEDatabaseView v1.62, https://www.nirsoft.net/utils/ese_database_view.html

[20] R. Chen, (2014, March), How do I convert a SID between binary and string forms?, https://blogs.msdn.microsoft.com/oldnewthing/20040315-00/?p=40253

[21] J. Metz, (2019, January), User Assist Key, https://github.com/libyal/winreg-kb/blob/master/documentation/User%20Assist%20keys.asciidoc

[22] J. Metz, (2019, January), Known Folder Identifiers, https://github.com/libyal/libfwsi/wiki/Known-Folder-Identifiers

[23] Sploit, (December 2012), SANS Forensic Artifact 6: UserAssist, http://sploited.blogspot.com/2012/12/sans-forensic-artifact-6-userassist.html

[24] D. Stevens, (January, 2019), UserAssist, https://blog.didierstevens.com/programs/userassist/