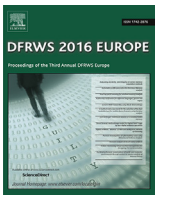
Digital Investigation 17 (2016) 1–13

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin



A forensic insight into Windows 10 Jump Lists

Bhupendra Singh*, Upasna Singh

Dept. of Computer Engineering, Defence Institute of Advanced Technology (DU), Girinagar, Pune, India

ARTICLE INFO

Article history:
Received 22 December 2015
Received in revised form 16 February 2016
Accepted 18 February 2016
Available online xxx

Keywords:
Jump lists
Windows forensics
Windows 10
LNK file analysis
DestList

ABSTRACT

The records maintained by Jump Lists have the potential to provide a rich source of evidence about users' historic activity to the forensic investigator. The structure and artifacts recorded by Jump Lists have been widely discussed in various forensic communities since its debut in Microsoft Windows 7. However, this feature has more capabilities to reveal evidence in Windows 10, due to its modified structure. There is no literature published on the structure of Jump Lists in Windows 10 and the tools that can successfully parse the Jump Lists in Windows 7/8, do not work properly for Windows 10. In this paper, we have identified the structure of Jump Lists in Windows 10 and compared it with Windows 7/8. Further, a proof-of-concept tool called JumpListExt (Jump List Extractor) is developed on the basis of identified structure that can parse Jump Lists in Windows 10, individually as well as collectively. Several experiments were conducted to detect anti-forensic attempts like evidence destruction, evidence modification and evidence forging carried out on the records of Jump Lists. Furthermore, we demonstrated the type of artifacts recorded by Jump Lists of four popular web browsers with normal and private browsing mode. Finally, the forensic capability of Jump Lists in Windows 10 is demonstrated in terms of activity timeline constructed over a period of time using Jump Lists.

Introduction

Jump Lists as a new feature was introduced in July 2009 with the release of Windows 7 and continued in later versions of Windows including Windows 10. Jump Lists are created by software applications or Operating System so that the user can "jump" directly to recently opened files and folders. The feature displays a limited number of items in Start or Taskbar, however, most of the recorded data is not displayed. In Windows 7/8, users can customize the number of Jump List items to be displayed by modifying the Registry value through 'regedit' application (Lyness, 2012). However, there is no way to change the number of Jump List items to be displayed in Windows 10, as it is hard coded.

Jump Lists maintain the records of recently accessed files and folders and group them as per application basis. Before the feature was introduced, forensic analysts had access to a short list of Most Recently Used (MRU) and Most Frequently Used (MFU) items in the Windows Registry (Lyness, 2012). Jump Lists can provide a lot more information about user's history and actions than MRU and MFU items. Jump Lists analysis can reveal useful information relating to files accesses, including the MRU and MFU list used by a user or application, file name, file path, MAC (Modified, Accessed, and Created) timestamps, volume name from which the file was accessed, and the history of uploaded and downloaded files through web browsers. These artifacts persist even after the files and their target applications are removed from the system (Larson, 2011).

Since the release of Windows 7, the structure and artifacts recovered from Jump Lists have been widely discussed in various forensic communities. Lallie and Bains (2012) presented the structure of Jump Lists in Windows 7.

* Corresponding author. Tel.: +919673095708.
E-mail addresses: bhupendra_pcse14@diat.ac.in (B. Singh), upasnasingh@diat.ac.in (U. Singh).

http://dx.doi.org/10.1016/j.diin.2016.02.001
1742-2876/

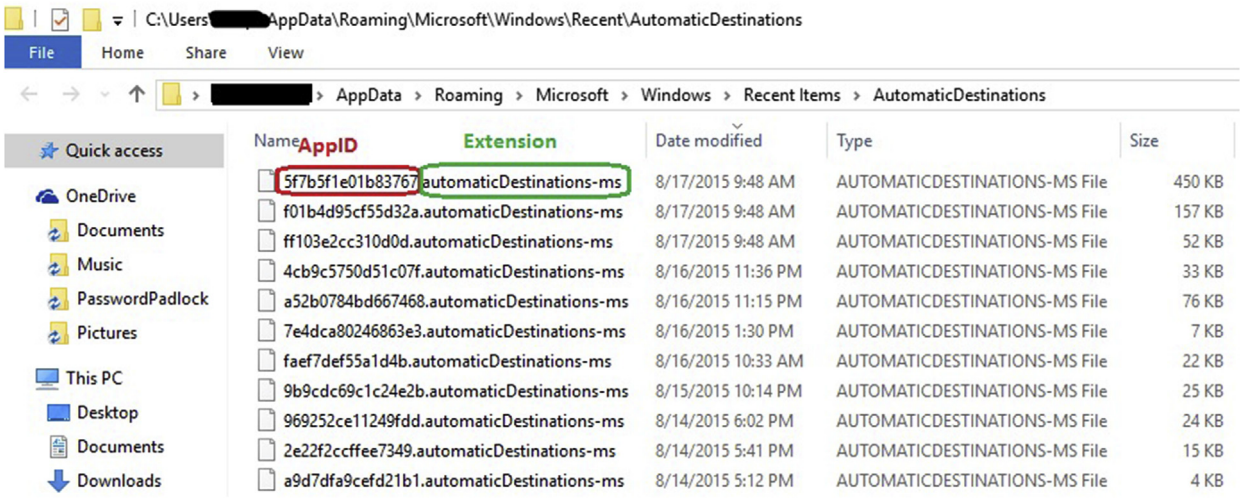


Fig. 1. Location of automaticDestinations-ms files in Windows 10.

Smith (2013) described a methodology to identify fraudulent documents using Jump Lists. There are many tools (Woan, 2013; MiTec, 2010; NirSoft, 2013; TZWorks, 2013) available to parse and view Jump Lists in Windows 7 and Windows 8, but none of them could successfully parse Jump Lists in Windows 10 as few fields in DestList stream are modified or added in Windows 10.

In this paper, we identified the new structure of Jump Lists in Windows 10 and compared it with existing ones. Further, a software tool, JumpListExt is developed to view and parse the Jump Lists based on the identified structure. Several experiments are carried out to demonstrate the importance of Jump Lists in a forensic investigation. The results show that a forensic timeline of user activities can be constructed by aggregating data extracted from all Jump Lists. The results also demonstrate that if any deletion or deliberate tampering is done in a Jump List file, then traces of such anti-forensic activities can be identified.

Motivation and research method

Though, there is limited literature published on the structure of DestList stream in Windows 7/8, we could not find any work related to new DestList stream structure in Windows 10. Also, none of the existing tools could successfully parse the Jump Lists in Windows 10.

For studying Jump Lists, the research method used in this paper is based on observations and experiments. For identifying the Jump List structure, different experiments were conducted and the identified structure is compared with the existing ones. Based on the identified structure, a proof-of-concept tool, called JumpListExt has been developed to parse the Jump Lists in Windows 10. Several experiments were conducted on four test computers: desktop, laptop and a virtual machine running Windows 10 Pro and other desktop running Windows 7 Professional. A number of activities on each computer were performed like installing and uninstalling applications; creating, deleting and accessing files with these applications; visiting websites, downloading and uploading files through web browsers and web searches using Cortana. These activities were performed on each computer in random order to create Jump List files. These test Jump Lists were then parsed using JumpListExt and conclusions drawn on the basis of observations are discussed in Experiments section. In addition to this, Jump Lists from several other computers were collected and parsed to evaluate the consistency of Jump List structure and to validate conclusions.

Identifying Windows 10 Jump List structure

Two types of Jump Lists are created within user's profile by Operating System or applications when a user performs certain actions. One is automaticDestinations-ms (autoDest) files in AutomaticDestinations subdirectory and the other is customDestinations-ms (custDest) files in CustomDestinations subdirectory. The full path to both types of Jump Lists are as follows:

- **AutoDest:** %UserProfile%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations
- **CustDest:** %UserProfile%\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations

These are hidden files and clicking through the directory structure using Windows Explorer will not reveal them.

Table 1
New applications with their AppIDs in Windows 10.

AppID	Application name
5f7b5f1e01b83767	Quick Access
4cb9c5750d51c07f	Movies & TV (Windows Store App)
a52b0784bd667468	Photos (Windows Store App)
ae6df75df512bd06	Groove Music (Windows Store App)
f01b4d95cf55d32a	File Explorer Windows 8.1/10
9d1f905ce5044aee	Microsoft Edge Browser

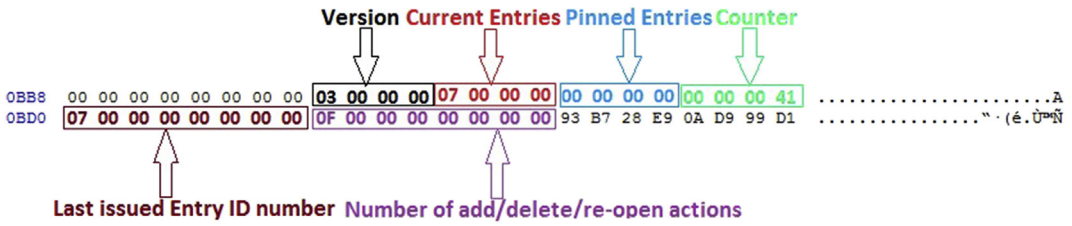


Fig. 2. DestList header structure in Windows 10.

These files can be viewed in Windows Explorer by entering the full path as the location. Fig. 1 shows the location of autoDest files in Windows 10.

Each Jump List file is named with a 16-digit hexadecimal number called AppID (Application Identifier) followed by an extension automaticDestinations-ms or customDestinations-ms. AppIDs are calculated by Windows OS for individual applications based on the file application path (Larson, 2011) and can be used to name the applications. So if the AppID is known then it can provide useful information for identifying the name of the application. Table 1 shows newly introduced applications with their AppIDs in Windows 10. The algorithm used to create the AppIDs is CRC-64 checksum obtained from the application path. It means that change in the path of an application to a non-default location would result in a different AppID (Lyness, 2012).

AutoDest Jump Lists

The autoDest files are Microsoft CFB (Compound File Binary) format files, also known as OLE (Object Linking and Embedding) files. AutoDest files are created by Windows OS and each file's name consists of unique AppID followed by extension an 'automaticDestinations-ms'. These files act as a container for individual hexadecimal numbered SHLLINK streams and a DestList stream. Each hexadecimal numbered stream follows the structure similar to Windows Shortcut (LNK) file. Thus, artifacts from any hexadecimal stream can be extracted and viewed using any LNK parser (Parsonage, 2010). The DestList stream, which follows a particular structure, records the order of file accesses and access count of each file that may serve as MRU and MFU list, respectively. The DestList stream structure in Windows 7 and Windows 8 is same, however, it has been modified in Windows 10.

DestList structure

To the best of our knowledge, there is no published information available regarding the structure of DestList stream in Windows 10 Jump Lists, at the time of writing this paper. Through various examinations conducted with a hex editor, we could confirm that the DestList stream structure in Windows 10 is different from Windows 7 and Windows 8. It was possible to figure out the major portions of DestList stream in Windows 10, however, the purpose of few areas remains unknown and may be identified further.

There are two portions of DestList stream: DestList header and DestList Entry. DestList header (32 bytes length) records the useful information such as total number of current Entries, last issue Entry ID number, pinned Entries, and number of added/deleted/re-opened actions. Fig. 2 shows hexadecimal view of DestList header in Windows 10 followed by Table 2 which shows the comparison of existing and identified DestList header structure.

Following the header, each Entry in the DestList has 130 bytes as opposed to 114 bytes in Windows 7/8, plus a variable length Unicode string data. Majority of the new Entry structure is same as in the earlier versions except few new fields and four additional null bytes added after variable length Unicode string data. In the new Entry structure, 4 bytes counter from offset 116 to 119 is added that records the access count of the individual Entries. The Entry having the largest access count signifies the MFU item.

Fig. 3 shows hexadecimal view of DestList Entry structure in Windows 10. In the example, the first 8 bytes (at offset 15,676) is checksum or hash of Entry that ensures the integrity of Entry data. If any change in Entry data (between the first byte and the last byte before the Entry string data)

Table 2
Comparison of existing and identified DestList header structure.

Offset	Size (bytes)	Description	
		DestList header structure in Windows 7/8 (Lyness, 2012; Lallie and Bains, 2012)	DestList header structure in Windows 10
0–3	4	Version number (value 1)	Version number (value 3)
4–7	4	Total number of current Entries	Total number of current Entries
8–11	4	Total number of pinned Entries	Total number of pinned Entries
12–15	4	Floating point value, some kind of counter	Unknown value
16–23	8	Last issued Entry ID number	Last Issued Entry ID number
24–31	8	Number of add/delete actions – increments as Entries are added and deleted	Number of add/delete/re-open actions – also increments as Entries are added, deleted and existing Entries are re-opened

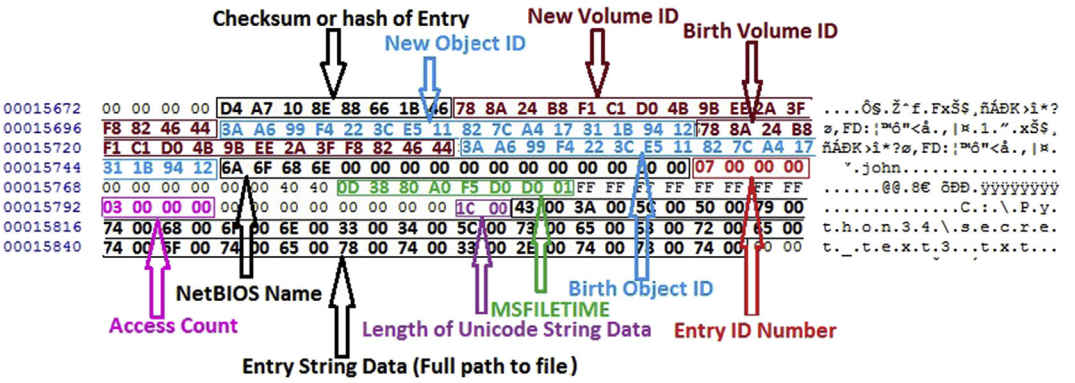


Fig. 3. DestList Entry structure in Windows 10.

is done then Entry would not appear in the list displayed in start/taskbar. Volume ID and Object ID are used to describe the file location at any point in time. By default NTFS embeds two file locations (New and Birth) and both values are same until file is moved from one volume to another. Windows OS creates 16 bytes Object ID to uniquely identify a file on a volume. The structure of Object ID can reveal the important artifacts such as timestamp of boot session (in GMT), sequence number (MFT Entry number) and MAC address of the primary network card in the computer (Parsonage, 2010). At offset of 16 bytes from 15,748, the location is reserved for NetBIOS name padded with null bytes. The next 4 bytes are reserved for Entry ID number which is used to ascertain the order in which Entries were added to list. At offset of 8 bytes from 15,776 signifies the timestamp (in GMT) of last recorded access of the Entry. At offset of 4 bytes from 15,792, is access count of the Entry which indicates the number of times the file was accessed. In the last portion of Entry structure is the variable length Entry string data followed by four null bytes that signifies full path to file. Table 3 shows the comparison of existing and identified DestList Entry structure.

CustDest Jump Lists

The custDest files are created by software applications and its name consists of unique AppID followed by an extension customDestinations-ms. These Jump Lists are specified by the applications using ICustomDestinationList API (Larson, 2011). As compared to autoDest files, these files follow different structure of sequential SHLLINK binary format. This type of Jump List files are limited in number and created mainly by web browsers and Windows Media Player. Web browsers create custDest files to record the user's web history on the system. For example, *969252ce11249fdd.customDestinations-ms* file is created by Mozilla Firefox that contains the timestamps and web history. Windows Media Player creates *74d7f43c1561fc1e.-customDestinations-ms* file that records file path of played music.

Designing parser for Windows 10 Jump Lists

This section describes the parsing method of both types of Jump Lists in Windows 10. A Python-based GUI tool named as JumpListExt[1] is developed based on this method, to parse individual Jump Lists. JumpListExt also has the functionality to parse and list aggregated data of all Jump Lists files in Windows 10. There are two modules in the source code,[2] one for decoding LNK stream and the other for decoding DestList stream.

Parsing AutoDest Jump List

As discussed in section 3, autoDest files are OLE files acting as a container for multiple SHLLINK streams and a DestList stream. Microsoft defined OLE files store multiple kind of objects within an object. For this, a simple filesystem like FAT is implemented within an OLE file. A single OLE file can be treated as a hierarchical collection of storage and stream objects that are analogous to directories and files, respectively (MSDN, 2015a). In Python 3.4, we can list all stream objects in an OLE file by importing *olefile* package. Below is the snippet of listing of all stream objects in an OLE file.

```
import olefile
ole = olefile.OleFileIO(''AppID.automaticDestinations-ms'', ''rb'')
print(ole.listdir(streams = True, storages = False))
```

The code lists all LNK streams and a DestList stream available in the given AppID. A LNK stream starts with 4 bytes signature 0x0000004C (decimal value 76) that

[1] https://sourceforge.net/projects/jumplistext/.
[2] https://sourceforge.net/p/jumplistext/code/ci/master/tree/.

Table 3
Comparison of existing and identified DestList Entry structure.

Offset	Size (bytes)	DestList entry structure in Windows 7/8 (Lyness, 2012; Lallie and Bains, 2012)	Offset	DestList entry structure in Windows 10	
		Description		Size (bytes)	Description
0–7	8	A checksum or hash of the Entry	0–7	8	A checksum or hash of the Entry
8–23	16	New Volume ID	8–23	16	New Volume ID
24–39	16	New Object ID	24–29	16	New Object ID
40–55	16	Birth Volume ID	40–55	16	Birth Volume ID
56–71	16	Birth Object ID	56–71	16	Birth Object ID
72–87	16	NetBIOS Name padded with zero to maximum length 16 bytes	71–87	16	NetBIOS Name padded with zero to maximum length 16 bytes
88–95	8	Entry ID number	88–91	4	Entry ID number
96–99	4	Floating point counter to record each time the file is accessed not necessarily opened	92–99	8	In all test '0x00 0x00 0x00 0x00'
100–107	8	MSFILETIME of last recorded access	100–107	8	MSFILETIME of last recorded access
108–111	4	Entry pin status '0xFF 0xFF 0xFF 0xFF' means unpinned, otherwise a counter starting at zero	108–111	4	Entry pin status '0xFF 0xFF 0xFF 0xFF' means unpinned, otherwise a counter starting at zero
112–113	2	Length of Unicode string data	112–115	4	In all tests, '0xFF 0xFF 0xFF 0xFF'
114-	variable	Entry string data	116–119	4	A counter that consistently increases as files are re-opened (access count)
			120–127	8	In all test '0x00 0x00 0x00 0x00'
			128–129	2	Length of Unicode string data
			130-	variable	Entry string data followed by '0x00 0x00 0x00 0x00'

represents the header size. DestList stream starts with 4 bytes version number which is 0x00000003 (decimal value 3) in Windows 10. Fig. 4 shows the flowchart for parsing an autoDest Jump List file.

Decoding LNK stream

LNK streams can be decoded using any available LNK parser as Microsoft has not modified its format in Windows 10. Also, the description of LNK file structure is provided by Microsoft in its Open Specifications Program for file formats (MSDN, 2015b). According to it, each LNK file consists of a header block of 76 bytes, location information block, data strings block and optional extra blocks. The developed tool decodes Modified, Accessed and Created timestamps (in UTC) from LNK header; Drive Type, Serial Number and Volume Name from location information block and LocalBasePath from data strings block. In addition to this, a LNK file may consist a link tracker properties data block that contains machine identifier, New and Birth Volume and Object IDs.

Volume and Object IDs fields are used to describe the location of a file at any point in time. Object ID is used to uniquely identify a file or directory on a volume. All Object IDs are stored on the volume and not maintained on FAT filesystem. When a file is created on a NTFS filesystem with an application, two Volume and two Object IDs (New and Birth) are embedded in the Jump List created by that application. These two file locations are same until the file is moved from one volume to another. Fig. 5 shows the two different Volume IDs when the file has been moved from one volume to other volume.

As shown in Fig. 5, after 16 bytes of New Volume ID, the next 16 bytes describe Object ID that is UUID (Universally Unique Identifier) specification which is created using the system time and system MAC address. Thus, decoding an Object ID can reveal these forensic artifacts. The structure of an Object ID is discussed in Table 4 with an example. The process of decoding of Object ID back to system time is same as given by Parsonage (2010).

Decoding DestList stream

If the value of first four bytes of the stream in the autoDest file is not 76, then it is DestList stream. DestList stream contains a header of 32 bytes followed by 130 bytes fixed structure plus a variable length Entry data for each Entry in DestList. In Windows 10, header starts with 0x00000003 (decimal value 3) that represents the version number. From the header, total number of current Entries and last issued Entry ID number can be extracted. Following the header, Entries are stored in sequential order following the structure presented in Table 3. The Volume and Object IDs are also created for each Entry in its structure and can be decoded as discussed above. We can iterate the process of decoding Entry data for each Entry present in DestList stream.

Parsing CustDest Jump List

This type of Jump List files follows a different structure of sequential SHLLINK streams. These SHLLINK streams can be extracted and parsed using the same module developed for decoding LNK stream in autodest file.

Aggregating data of all Jump Lists

The developed tool also has the functionality of parsing all Jump List files and listing the aggregated data on user interface. For aggregating the data, the tool first parses all

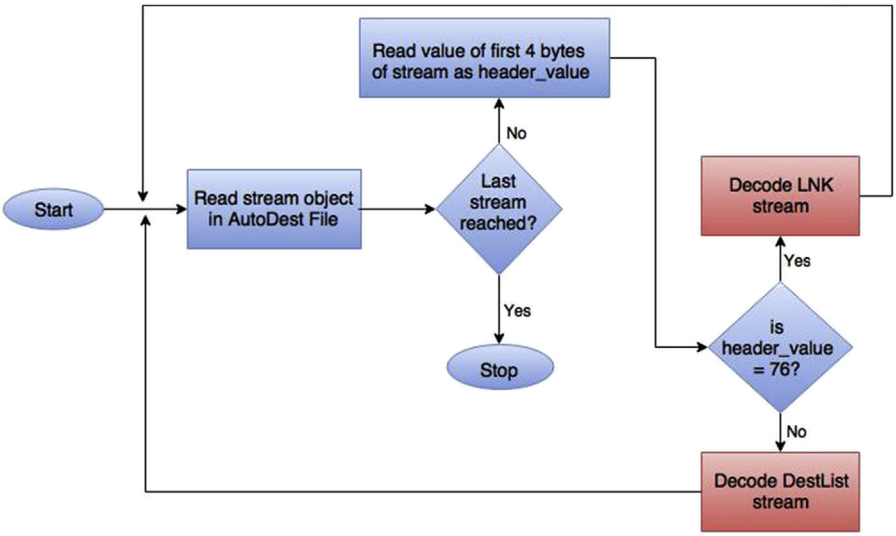


Fig. 4. A Flowchart for parsing an autoDest Jump List.

autoDest files under the AutomaticDestinations directory and then parses all custDest files under the CustomDestinations directory. This parsed data from both type of files is then aggregated and listed on user interface. If we parse all files, and select 'Export' option, the tool creates a SQLite database named as *JumpListData.sqlite* containing aggregated data in a table named *Jump*. The schematic diagram of tool for aggregating data is shown in Fig. 6.

Experiments

Experiments were conducted in three main categories, namely, for detection of anti-forensic attempts carried out on the Jump Lists; for identifying the type of information recorded in Jump Lists of different web browsers in normal and private browsing mode and finally to assimilate a timeline of user activities over a period of time using Jump Lists. The detection of anti-forensic attempts have been demonstrated for three possible cases, i.e., evidence destruction (deleting Entries from Jump List), evidence modification (modifying content of Jump List) and evidence forging (forging Jump Lists of Windows 10 by Jump Lists of Windows 7/8). The following subsections demonstrate the said experimental results.

Deleting entries from Jump List

For each Entry in autoDest Jump List, an Entry ID number is reserved in its DestList Entry structure. DestList header contains total number of current Entries and the last

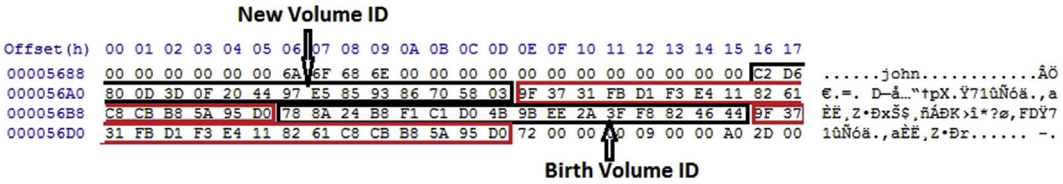


Fig. 5. Volume and Object IDs of a LNK stream in an autoDest Jump List.

Table 4
Object ID structure.

Object ID	9F 37 31 FB D1 F3 E4 11 82 61 C8 CB B8 5A 95 D0
Lower order time	9F 37 31 FB
Middle order time	D1 F3
Higher order time and version	E4 11
Sequence number	82 61
MAC address	C8 CB B8 5A 95 D0

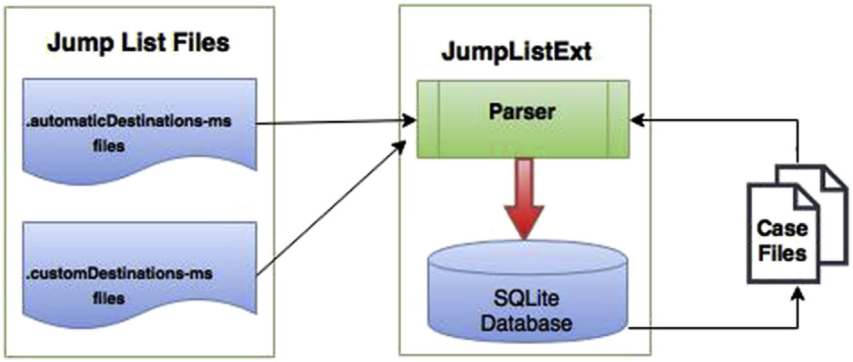


Fig. 6. Schematic diagram of aggregating data extracted from all Jump Lists.

Fig. 7. DestList header of Microsoft Word 2013 Jump List before deletion.

issued Entry ID number to keep track of allocated Entries in Jump List. In the last 8 bytes of DestList header, there is a counter that increments by two if a new Entry is created. However, counter increments by one if an existing Entry is opened or removed from the Jump List. Any Entry from the Jump List can be removed by right clicking the mouse and selecting 'Remove from this list' option. If an Entry is deleted from the Jump List, the deletion can be detected in two ways:

- If the last issued Entry ID number is not equal to the total number of current Entries in the Jump List.
- If one or more values in the numerical sequence of issued Entry ID number is missing.

Fig. 7 shows the DestList header of Microsoft Word 2013 Jump List before deletion. It can be deduced that no Entry is removed from the Jump List as total number of current Entries and the last issued Entry ID number are same. From Figure, it is also evident that few Entries are accessed more than one time (access count >1) because if all 8 Entries were accessed only once then the counter value (number of add/delete/re-open actions) would have been $2 \times 8 = 16$ while it is 39. We parsed this Jump List file (see Fig. 8) and the access count of all Entries in DestList was added to be 31. If the counter value is Z, the last issued Entry ID number is L, then it was observed that the following equation will hold:

$$Z = L + \sum_{en=1}^{en=L} Access\_Count_{en} \quad (1)$$

Where $Access\_Count_{en}$ is access count of Entry ID number en.

To see the differences in the DestList header, we removed the Entry ID number 6 from the Word 2013 Jump List. Fig. 9 shows the DestList header after deleting the said Entry.

It can be observed that the counter value is incremented by one after deletion and total number of current Entries

DestList (File Name: 47bb2136fda3f1ed.automaticDestinations-ms, Application Name: Microsoft Office Word 2013 x64)

E.I	Netbi	Last Recorded Access	Access	New (Timestamp)	Seq. N	Birth (Mac)	Data
3	john	25 Nov 2015 10:02:50 AM	2	12 Nov 2015 04:26:41 AM	33522	a4:17:31:1b:94:12	E:\Modified_DIARA_Proposal.docx
8	john	25 Nov 2015 10:00:10 AM	1	08 Nov 2015 04:45:22 AM	33516	a4:17:31:1b:94:12	E:\SANDISK\Notesheet_for_LAN_in_DFL.docx
7	john	25 Nov 2015 09:56:15 AM	2	06 Nov 2015 04:57:23 AM	33512	a4:17:31:1b:94:12	C:\Users\bhupi\Desktop\Windows 10 JumpList Image\AppIDs.docx
6	john	25 Nov 2015 09:55:02 AM	1	10 Aug 2015 04:05:58 AM	33410	a4:17:31:1b:94:12	C:\Users\bhupi\Desktop\final_jumplist_w10.docx
5	john	25 Nov 2015 09:52:50 AM	1	17 Nov 2015 04:22:29 AM	33531	a4:17:31:1b:94:12	C:\Users\bhupi\Desktop\JDFSL JumpList Paper\Forensic Implications of Jump L
4	john	25 Nov 2015 09:52:27 AM	1	16 Apr 2015 08:30:40 AM	33372	c8:cb:b8:5a:95:d0	D:\CDAC\CDAC Cyber Forensics worksop on 16 April 2015\cdac\Installation Set
2	john	25 Nov 2015 09:51:43 AM	1	17 Nov 2015 04:22:29 AM	33531	a4:17:31:1b:94:12	C:\Users\bhupi\Desktop\JDFSL JumpList Paper\JDFSL-Template-2014.docx
1	john	24 Nov 2015 02:28:35 PM	22	12 Nov 2015 04:26:41 AM	33522	a4:17:31:1b:94:12	C:\Users\bhupi\Desktop\JDFSL JumpList Paper\dateconvertormethod.docx

Fig. 8. Parsed output of Microsoft Word 2013 Jump List.

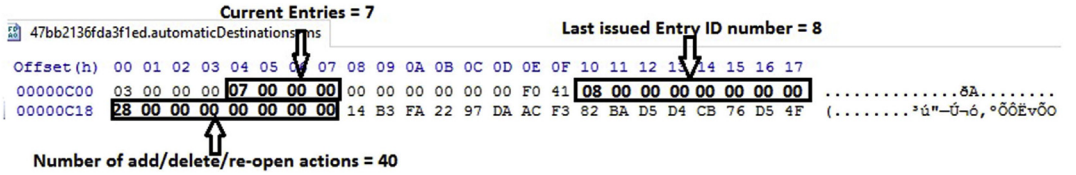


Fig. 9. DestList header of Microsoft Word 2013 Jump List after deletion.

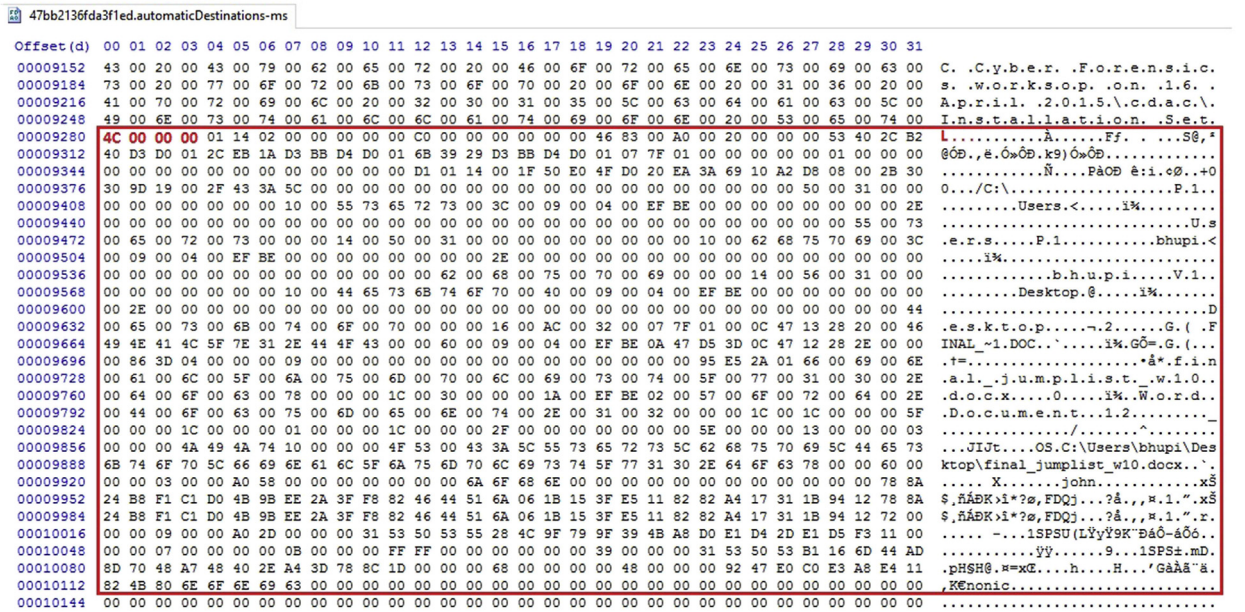


Fig. 10. Before deletion of an Entry from the Jump List.

are now 7. If few Entries are removed from the Jump List, then number of deleted Entries (say D) can be calculated as $D = L - C$ where C is number of current Entries in the Jump List and access count of the deleted Entries can be calculated using the Equation (2).

$$Z = L + \sum_{en=1}^{en=C} Access\_Count_{en} + D \tag{2}$$

Carving the deleted entries

We observed that when an Entry is removed from the Jump List, the size of DestList stream is reduced by the size of deleted Entry structure (130 bytes + length of Unicode string data). However, whole or part of LNK stream of the deleted Entry may still be present in Jump List. If whole LNK stream of the deleted Entry is available in the Jump List, then it can be carved by saving the LNK stream to a separate file with extension LNK, using a hex editor. A Python-based *LNKParser.py* script[3] was developed for parsing the extracted LNK files. If multiple Entries are removed from the Jump List, the process can be repeated to carve the individual deleted Entries. Fig. 10, shows the LNK stream of the Entry ID number 6 before the Entry was removed. The size of the stream is 864 bytes starting from 4 bytes signature 0x0000004 C at offset 9280 to offset 10,143.

Fig. 11 shows the LNK stream of Entry number 6 after removal of Entry. File header, link target identifier and location information elements are overwritten with the data of existing Entries. However, the last 448 bytes are still present in the Jump List including the data strings, distributed link tracker data properties and extra blocks. Since the LNK header is missing, the available blocks in LNK stream can be parsed manually.

Data string block in LNK file format records the LocalBasePath which signifies the basepath to target file. The next link tracker block starts with 4 byte signature 0x00000060 (decimal value 96) containing 16 bytes for machine identifier, 16 bytes for New Volume ID, 16 bytes for New Object ID, 16 bytes for Birth Volume ID and 16 bytes for Birth Object ID. From Volume and Object IDs, the timestamp of boot session, sequence number and MAC address of the system can be extracted as discussed in Designing parser for Windows 10 Jump Lists. Thus, we can detect and identify the deletions and can calculate the access

[3] https://sourceforge.net/p/jumplistext/code/ci/master/tree/.

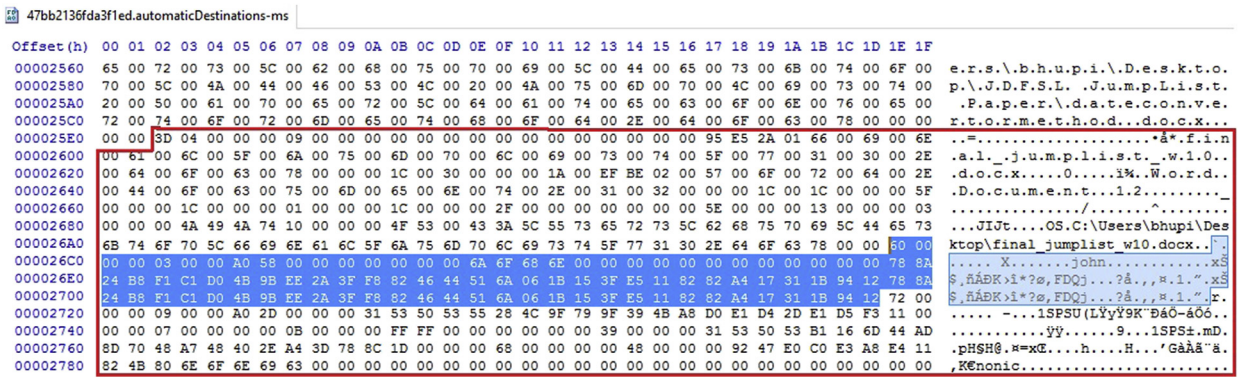


Fig. 11. After deletion of the Entry from the Jump List.

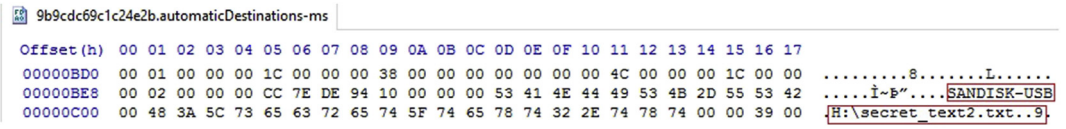


Fig. 12. Before modification of Volume name and LocalBasePath.

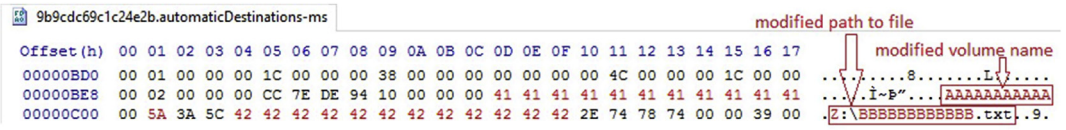


Fig. 13. After modification of Volume name and LocalBasePath.

count of the deleted Entries using Equation (2). However, complete carving of the deleted Entries from Jump Lists is still a forensic research challenge.

Modifying content of Jump List

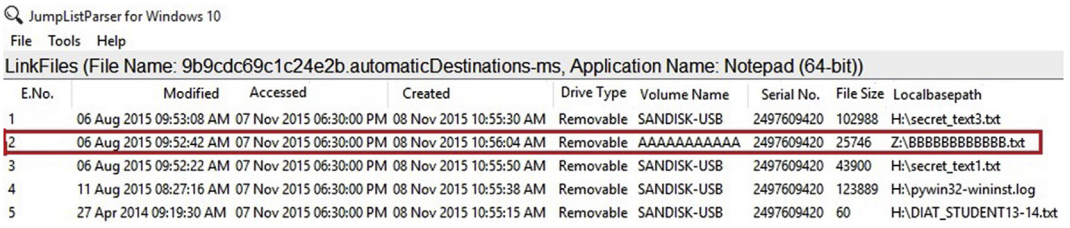
This experiment was conducted to simulate the condition in which the user of suspected system modifies the content of an autoDest Jump List in order to produce false evidence. The modifications in the content of the Jump List can be done with the help of any hex editor. Two different experiments were performed to detect the modifications carried out in the content of a LNK stream and DestList stream. For this, Notepad application was considered and new text files were created and few existing text files were opened from both fixed and removable media devices, thus creating *9b9cdc69c1c24e2b.automaticDestinations-ms* Jump List. In first experiment, the content of any LNK stream in this Jump List was modified. The Fig. 12 shows the Volume name as *SANDISK-USB* and LocalBasePath as *H:\secret_text2.txt* present in LNK stream. Now the Volume name and LocalBasePath is modified as shown in Fig. 13. In second experiment, the content of DestList stream was modified.

It was observed that when any modification in the content of the any Jump List is done, Windows creates a backup copy of that Jump List file with the same name followed by bak extension. Modifying the LNK stream content does not affect list displayed in start/taskbar even if we change the whole data (except 4 bytes signature) of individual LNK streams. The snapshot of parsed data using JumpListExt, shown in Fig. 14 is from modified *9b9cdc69c1c24e2b.automaticDestinations-ms* file. It was also observed that modifications done are not persistent and exist until the file is accessed again. Also, the modification traces are evidentiary if the whole LNK stream is modified then it can be easily detected by the DestList data of the same Entry ID number. However, if the part of the data in LNK stream which cannot be constructed from DestList stream, is modified then such modifications can only be detected by moving to target file location and validating the metadata with the parsed metadata.

If we change any byte of the Entry data in DestList stream except the data string, then that item is not displayed due to mismatch in checksum or hash value. Further, if any new files are accessed by the considered application after such modifications, then no new Entries will be added in the Jump List. This provides the requisite mechanism to identify the modifications in the DestList stream.

Forging Jump Lists of Windows 10 by Jump Lists of Windows 7/8

This experiment was conducted to detect the condition in which a user of suspected system replaces its Jump Lists



JumpListParser for Windows 10

File Tools Help

LinkFiles (File Name: 9b9cdc69c1c24e2b.automaticDestinations-ms, Application Name: Notepad (64-bit))

E.No.	Modified	Accessed	Created	Drive Type	Volume Name	Serial No.	File Size	Localbasepath
1	06 Aug 2015 09:53:08 AM	07 Nov 2015 06:30:00 PM	08 Nov 2015 10:55:30 AM	Removable	SANDISK-USB	2497609420	102988	H:\secret_text3.txt
2	06 Aug 2015 09:52:42 AM	07 Nov 2015 06:30:00 PM	08 Nov 2015 10:56:04 AM	Removable	AAAAAAAAAAA	2497609420	25746	Z:\BBBBBBBBBBB.txt
3	06 Aug 2015 09:52:22 AM	07 Nov 2015 06:30:00 PM	08 Nov 2015 10:55:50 AM	Removable	SANDISK-USB	2497609420	43900	H:\secret_text1.txt
4	11 Aug 2015 08:27:16 AM	07 Nov 2015 06:30:00 PM	08 Nov 2015 10:55:38 AM	Removable	SANDISK-USB	2497609420	123889	H:\pywin32-wininst.log
5	27 Apr 2014 09:19:30 AM	07 Nov 2015 06:30:00 PM	08 Nov 2015 10:55:15 AM	Removable	SANDISK-USB	2497609420	60	H:\DIAT_STUDENT13-14.txt

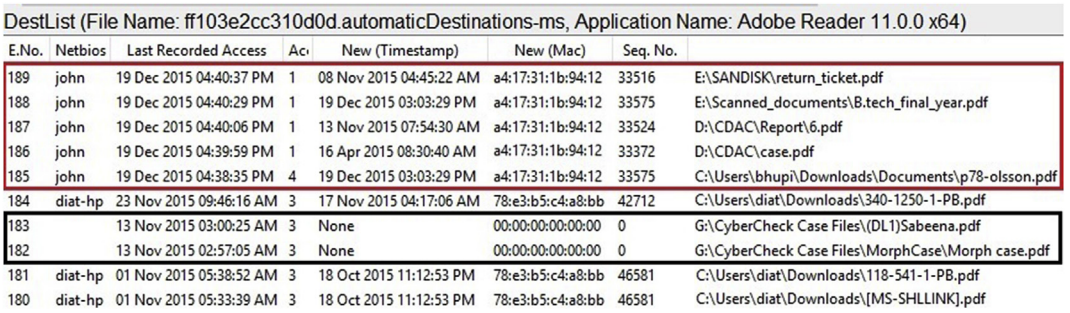
Fig. 14. Parsed output after modifying the content of a Jump List.

from any benign system running same or different distributions of Windows OS, in order to mislead the forensic investigator. In this case, same version of an application (Adobe Reader 11.0) was installed at its default directory on two systems (one suspected and other benign) running different distributions of Windows OS. The suspected system was running Windows 10 Pro with computer name "john" while the other system was running Windows 7 Professional with computer name "diat-hp". On both the systems, a series of activities were performed randomly by creating new *pdf* files and opening the existing files with this installed application to create *ff103e2cc310d0d.automaticDestinations-ms* file. This autodest Jump List file in suspected system was then replaced with the same Jump List created in Windows 7 system. Subsequently, few more operations were performed with the considered application in suspected system to see the changes in the replaced Jump List file, which was then parsed through JumpListExt for analysis.

Parsing LNK stream or DestList stream exposes the machine name and system MAC address, as Volume and Object IDs are maintained for each accessed file in LNK stream and DestList stream of Jump List. Fig. 15 shows the DestList data of the *ff103e2cc310d0d.automaticDestinations-ms* file with two different NetBIOS name. This information can easily reveal the possible forgery of Jump Lists in a suspected system. However, if the files are accessed from removable media devices (FAT filesystem) then it is difficult to identify forgery from machine name and MAC address as Volume and Object IDs are not maintained on FAT filesystem. Further, was observed that no item is displayed in start/taskbar of the suspected system until we perform at least one operation with the application under consideration. However, in a different experiment with both systems running same OS (Windows 10 Pro), replacing the Jump List of an application in system1 with Jump List of same name in system2, displayed the accessed items with that application in system1, without performing such actions.

Information recorded in Jump Lists of different web browser

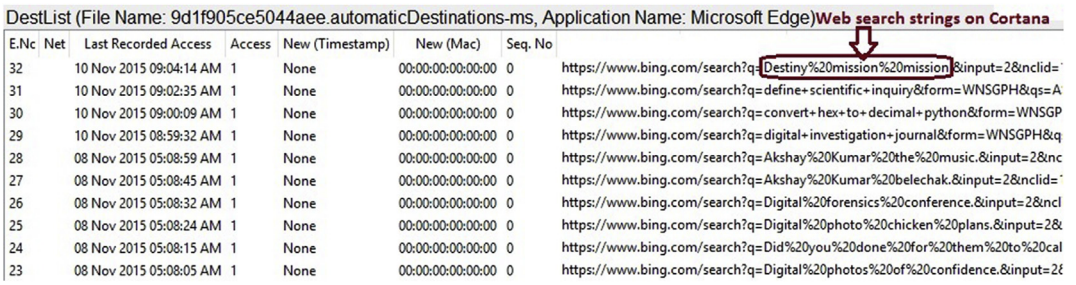
Cortana is one of the new features introduced in Windows 10. A user can perform text or voice search on local computer or on web using this feature. If a web search is performed using Cortana, then search is directed to the default web browser in the system. For example, if the default web browser is Microsoft Edge then the search results are obtained by Bing search engine. Otherwise the search results are obtained by the search engine of the default web browser. The Jump List created by any browser used as default in Windows 10 always records the URL address and timestamp of web search in its DestList stream. Two different experiments were carried out to test the type of information recorded in four popular browsers' Jump Lists. In first experiment, Microsoft Edge browser was selected as the default web browser and Google as default search engine. Random text and voice searches were performed using Cortana followed by downloading and uploading of files, thus creating *9d1f905ce5044aee.automaticDestinations-ms* file. It was observed that for each web search, only the Entry ID number, the last recorded access (timestamp of search), access count (always 1 for each Entry) and Data (URL located in the address bar) are recorded in the DestList stream of the Jump List file. Fig. 16 shows the search string and related artifacts after parsing the Jump List of Microsoft Edge browser. No information was recorded related to downloaded or uploaded files in the Jump List.



DestList (File Name: ff103e2cc310d0d.automaticDestinations-ms, Application Name: Adobe Reader 11.0.0 x64)

E.No.	Netbios	Last Recorded Access	Ac	New (Timestamp)	New (Mac)	Seq. No.	
189	john	19 Dec 2015 04:40:37 PM	1	08 Nov 2015 04:45:22 AM	a4:17:31:1b:94:12	33516	E:\SANDISK\return_ticket.pdf
188	john	19 Dec 2015 04:40:29 PM	1	19 Dec 2015 03:03:29 PM	a4:17:31:1b:94:12	33575	E:\Scanned_documents\B.tech_final_year.pdf
187	john	19 Dec 2015 04:40:06 PM	1	13 Nov 2015 07:54:30 AM	a4:17:31:1b:94:12	33524	D:\CDAC\Report\6.pdf
186	john	19 Dec 2015 04:39:59 PM	1	16 Apr 2015 08:30:40 AM	a4:17:31:1b:94:12	33372	D:\CDAC\case.pdf
185	john	19 Dec 2015 04:38:35 PM	4	19 Dec 2015 03:03:29 PM	a4:17:31:1b:94:12	33575	C:\Users\bhupi\Downloads\Documents\p78-olsson.pdf
184	diat-hp	23 Nov 2015 09:46:16 AM	3	17 Nov 2015 04:17:06 AM	78:e3:b5:c4:a8:bb	42712	C:\Users\diat\Downloads\340-1250-1-PB.pdf
183		13 Nov 2015 03:00:25 AM	3	None	00:00:00:00:00:00	0	G:\CyberCheck Case Files\(DL1)Sabeena.pdf
182		13 Nov 2015 02:57:05 AM	3	None	00:00:00:00:00:00	0	G:\CyberCheck Case Files\MorphCase\Morph case.pdf
181	diat-hp	01 Nov 2015 05:38:52 AM	3	18 Oct 2015 11:12:53 PM	78:e3:b5:c4:a8:bb	46581	C:\Users\diat\Downloads\118-541-1-PB.pdf
180	diat-hp	01 Nov 2015 05:33:39 AM	3	18 Oct 2015 11:12:53 PM	78:e3:b5:c4:a8:bb	46581	C:\Users\diat\Downloads\[MS-SHLLINK].pdf

Fig. 15. NetBIOS and MAC address displayed in Jump List.


DestList (File Name: 9d1f905ce5044aee.automaticDestinations-ms, Application Name: Microsoft Edge) **Web search strings on Cortana**

E.Nc	Net	Last Recorded Access	Access	New (Timestamp)	New (Mac)	Seq. No	
32		10 Nov 2015 09:04:14 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=Destiny%20mission%20mission&input=2&nclid=
31		10 Nov 2015 09:02:35 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=define+scientific+inquiry&form=WNSGPH&qs=A
30		10 Nov 2015 09:00:09 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=convert+hex+to+decimal+python&form=WNSGP
29		10 Nov 2015 08:59:32 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=digital+investigation+journal&form=WNSGPH&q
28		08 Nov 2015 05:08:59 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=Akshay%20Kumar%20the%20music.&input=2&nc
27		08 Nov 2015 05:08:45 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=Akshay%20Kumar%20belechak.&input=2&nclid=
26		08 Nov 2015 05:08:32 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=Digital%20forensics%20conference.&input=2&ncl
25		08 Nov 2015 05:08:24 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=Digital%20photo%20chicken%20plans.&input=2&
24		08 Nov 2015 05:08:15 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=Did%20you%20done%20for%20them%20to%20cal
23		08 Nov 2015 05:08:05 AM	1	None	00:00:00:00:00:00	0	https://www.bing.com/search?q=Digital%20photos%20of%20confidence.&input=2&

Fig. 16. Artifacts recorded in Jump List of Microsoft Edge browser.

In second experiment, Mozilla Firefox 42.0 was installed and selected as default web browser. Again, random text and voice searches were performed using Cortana followed by downloading and uploading of files which created *969252ce11249fdd.automaticDestinations-ms* file. It was observed that this Jump List file contained information related to downloaded and uploaded files along with the search strings (see Fig. 17). However, it is difficult to distinguish between downloaded and uploaded files from the Jump List. Also, if the files are downloaded with any download manager, information related to such files are not stored in browser's Jump List.

A comparison based on above methodology was conducted on four web browsers, namely, Microsoft Edge, Mozilla Firefox 42.0, Google Chrome 47.0 and Opera 34.0. The results of the recorded artifacts in Jump Lists of these web browsers are shown in Table 5.

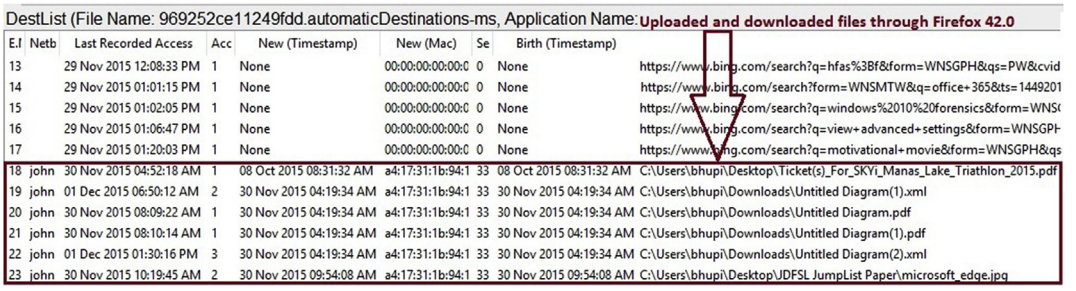
Aggregating data of all Jump Lists for construction of activity timeline

Jump Lists contains a treasure of information for creating different timelines of the activities performed on a system. Through Jump Lists analysis, forensic investigator can know the accessed, created, modified files and their access count between the two time instances. This experiment was performed to construct a timeline of user activities performed on a computer running Windows 10 Pro for a period of 10 days starting from 01 Dec 2015 12:00:00 AM to 10 Dec 2015 12:00:00 AM. For this, several applications were installed and number of files were created, accessed and modified with preinstalled and user installed applications, in random order, resulting in creation of a Jump List for each application. For creating the timeline, JumpListExt was run and "Parse all" option was selected, which created a SQLite database named *JumpListData.sqlite* containing aggregated data extracted from all Jump Lists. This database consists a table with name *Jump* having 14 columns with header name *AppID*, *ENo*, *NetBIOS Name*, *Last Recorded Access*, *Modified*, *Accessed*, *Created*, *Access Count*, *DriveType*, *VolumeName*, *BirthTimestamp*, *BirthMAC*, *SeqNo* and *Data* for each Entry in a Jump List. Fig. 18 shows the aggregated data of all Jump Lists containing only seven (out of 14) columns, for visual convenience.

It can be seen from Fig. 18 that the system was started at 9:31:57 AM on 03 Dec 2015 (refer column name Birth). Subsequently, file name indicated as 1 in Fig. 18 was accessed at 9:43:07 AM on the same day (refer column name Last Recorded Access) which was accessed for the 2*nd* time (refer column name Access Count). After this, file name indicated as 2 in Fig. 18 was created at 12:15:28 PM, followed by accessing of file name indicated as 3 in Fig. 18 at 12:50:20 PM (refer column name Last Recorded Access), which was accessed for the 3*rd* time (refer column name Access Count). In this way the activity timeline of 10 days was created and is shown in Table 6.

Discussion of results

The results of first experiment, aimed at detecting anti-forensic attempts, demonstrate that if few Entries are deleted from a Jump List then it can be easily detected and identified. Also, part of the deleted Entries can be carved and parsed manually. An equation is established on the


DestList (File Name: 969252ce11249fdd.automaticDestinations-ms, Application Name: **Uploaded and downloaded files through Firefox 42.0**

E.I	Netb	Last Recorded Access	Acc	New (Timestamp)	New (Mac)	Se	Birth (Timestamp)	
13		29 Nov 2015 12:08:33 PM	1	None	00:00:00:00:00:0	0	None	https://www.bing.com/search?q=hfas%3Bf&form=WNSGPH&qs=PW&cvid
14		29 Nov 2015 01:01:15 PM	1	None	00:00:00:00:00:0	0	None	https://www.bing.com/search?form=WNSMTW&q=office+365&ts=1449201
15		29 Nov 2015 01:02:05 PM	1	None	00:00:00:00:00:0	0	None	https://www.bing.com/search?q=windows%2010%20forensics&form=WNS(
16		29 Nov 2015 01:06:47 PM	1	None	00:00:00:00:00:0	0	None	https://www.bing.com/search?q=view+advanced+settings&form=WNSGPH
17		29 Nov 2015 01:20:03 PM	1	None	00:00:00:00:00:0	0	None	https://www.bing.com/search?q=motivational+movie&form=WNSGPH&qs
18	john	30 Nov 2015 04:52:18 AM	1	08 Oct 2015 08:31:32 AM	a4:17:31:1b:94:1	33	08 Oct 2015 08:31:32 AM	C:\Users\bhupi\Desktop\Ticket(s)_For_SKYi_Manas_Lake_Triathlon_2015.pdf
19	john	01 Dec 2015 06:50:12 AM	2	30 Nov 2015 04:19:34 AM	a4:17:31:1b:94:1	33	30 Nov 2015 04:19:34 AM	C:\Users\bhupi\Downloads\Untitled Diagram(1).xml
20	john	30 Nov 2015 08:09:22 AM	1	30 Nov 2015 04:19:34 AM	a4:17:31:1b:94:1	33	30 Nov 2015 04:19:34 AM	C:\Users\bhupi\Downloads\Untitled Diagram.pdf
21	john	30 Nov 2015 08:10:14 AM	1	30 Nov 2015 04:19:34 AM	a4:17:31:1b:94:1	33	30 Nov 2015 04:19:34 AM	C:\Users\bhupi\Downloads\Untitled Diagram(1).pdf
22	john	01 Dec 2015 01:30:16 PM	3	30 Nov 2015 04:19:34 AM	a4:17:31:1b:94:1	33	30 Nov 2015 04:19:34 AM	C:\Users\bhupi\Downloads\Untitled Diagram(2).xml
23	john	30 Nov 2015 10:19:45 AM	2	30 Nov 2015 09:54:08 AM	a4:17:31:1b:94:1	33	30 Nov 2015 09:54:08 AM	C:\Users\bhupi\Desktop\JDFSL JumpList Paper\microsoft_edge.jpg

Fig. 17. Artifacts recorded in Jump List of Mozilla Firefox 42.0 web browser.

Table 5
A comparison of recorded artifacts in Jump Lists of different web browsers.

Browsers	Web search strings on Cortana		Uploading/downloading a file	
	Normal browsing mode	Private browsing mode	Normal browsing mode	Private browsing mode
Microsoft Edge	✓	✓	×	×
Mozilla Firefox 42.0	✓	✓	✓	×
Google Chrome 47.0	✓	✓	×	×
Opera 34.0	✓	✓	×	×

basis of observations to calculate the access count of the deleted Entries. The results of second experiment, for detection of modified Entries, demonstrate that deliberate tampering with the Jump List data can be detected by validating the data of an Entry in LNK stream with the data of the same Entry in DestList stream. However, if tampering is done with the data field of an Entry that is not stored in DestList like MAC timestamps, then it is difficult to detect such deliberate tampering from Jump List. The result of third experiment, for detection of forged Entries, demonstrates that if the Jump Lists with the same name, created by the same application running on two different computers (with same or different Windows OS), then the Jump List analysis can reveal forgery based on computer name and unique MAC address. The results of the fourth experiment show that web searches performed by different web browsers, using Cortana, can be extracted for forensic analysis by parsing the Jump Lists created by these web browsers. Further, it was observed during this experiment that the Jump List of Mozilla Firefox 42.0 (one of the four web browsers considered for analysis) also records the artifacts related to uploaded and downloaded files in normal browsing mode. The results of last experiment demonstrate that the aggregated data created by parsing all Jump Lists can prove to be very useful for construction of activity timelines. For this, a SQLite database as the case file of aggregated data is created while parsing all Jump List files, which can be used by investigators to filter this data based on requirement.

Limitations

The developed software tool presented in this paper focuses on parsing and displaying the parsed data on user interface. We can also export the parsed data into a SQLite database by selecting the 'Export' button in File menu of software. The software tool can be extended to run SQL query on the created database from user interface so that the investigators can filter and list the data based on requirement. Future endeavors would be to enable automation for user defined analysis of the parsed data.

Conclusion

Jump Lists analysis can provide detailed forensic insight into users' activities that can be utilized for forensic analysis to construct a timeline of activities. Jump Lists provide artifacts pertaining to file creation, access, modification or deletion which can be of immense importance to a forensic investigator in cases when user's activity history is of investigator's interest.

In this paper, the Jump List structure of Windows 10 was identified and compared with Windows 7/8. The newly added or modified field values in DestList stream reveal more information for forensic analysis. Based on the identified structure, a software tool has been developed for parsing Jump Lists, individually as well as collectively. The results of the experiments demonstrate that if evidence

JumpListExt for Windows 10

File Tools Help

Last Recorded Access	Modified	Accessed	Created	Acc	Drive Type	Volume Name	Birth(Timestamp)	
03 Dec 2015 09:43:07 AM	28 Nov 2015 06:44:44 PM	28 Nov 2015 06:44:44 PM	28 Nov 2015 06:44:44 PM	2	Fixed	OS	03 Dec 2015 09:31:57 AM	1 C:\Users\bhupi\Desktop\
03 Dec 2015 12:15:28 PM	03 Dec 2015 12:15:27 PM	03 Dec 2015 12:15:27 PM	03 Dec 2015 12:15:27 PM	1	Fixed	OS	03 Dec 2015 09:31:57 AM	2 C:\Users\bhupi\AppData
03 Dec 2015 12:50:20 PM	23 Apr 2015 12:48:03 AM	04 Jun 2015 04:58:36 PM	04 Jun 2015 04:58:36 PM	3	Fixed	softwares	03 Dec 2015 09:31:57 AM	3 D:\PhD@DIAT\PhD@2nc
06 Dec 2015 12:04:04 PM	22 Apr 2015 09:15:23 PM	22 Apr 2015 09:15:23 PM	22 Apr 2015 09:15:23 PM	1	Fixed	softwares	06 Dec 2015 11:58:42 AM	4 D:\PhD@DIAT\compreh
07 Dec 2015 09:13:48 AM	07 Dec 2015 09:13:47 AM	07 Dec 2015 09:13:47 AM	07 Dec 2015 09:13:47 AM	1	Fixed	OS	07 Dec 2015 08:58:05 AM	5 C:\Users\bhupi\Downloa
07 Dec 2015 11:22:12 AM	07 Dec 2015 11:20:50 AM	06 Dec 2015 06:30:00 PM	07 Dec 2015 11:20:49 AM	2	Removable	SANDISK-USB	07 Dec 2015 08:58:05 AM	6 H:\Digital investigation P
07 Dec 2015 11:31:04 AM	07 Dec 2015 11:29:04 AM	12 Dec 2015 07:05:23 AM	07 Dec 2015 11:29:03 AM	2	Fixed	softwares	07 Dec 2015 08:58:05 AM	7 D:\PhD@DIAT\Jump List
07 Dec 2015 11:42:27 AM	07 Dec 2015 11:42:21 AM	07 Dec 2015 11:42:19 AM	07 Dec 2015 11:42:19 AM	1	Fixed	OS	07 Dec 2015 08:58:05 AM	8 C:\Users\bhupi\Downloa
10 Dec 2015 10:03:32 AM	10 Dec 2015 10:01:16 AM	10 Dec 2015 10:00:31 AM	10 Dec 2015 10:00:31 AM	2	Fixed	OS	10 Dec 2015 09:50:58 AM	9 C:\Users\bhupi\Downloa
11 Dec 2015 04:02:15 AM	10 Dec 2015 07:25:56 AM	11 Dec 2015 04:01:10 AM	11 Dec 2015 04:01:10 AM	1	Fixed	OS	11 Dec 2015 03:53:27 AM	C:\Users\bhupi\Desktop\
11 Dec 2015 10:12:16 AM	30 Jan 2012 07:14:28 AM	11 Dec 2015 09:19:53 AM	11 Dec 2015 09:19:53 AM	4	Fixed	OS	11 Dec 2015 03:53:27 AM	C:\Users\bhupi\Desktop\
11 Dec 2015 09:21:35 AM	11 Dec 2015 09:05:54 AM	11 Dec 2015 09:19:55 AM	11 Dec 2015 09:19:55 AM	1	Fixed	OS	11 Dec 2015 03:53:27 AM	C:\Users\bhupi\Desktop\
11 Dec 2015 10:27:08 AM	11 Dec 2015 10:27:08 AM	11 Dec 2015 10:27:08 AM	11 Dec 2015 09:21:16 AM	11	Fixed	OS	11 Dec 2015 03:53:27 AM	C:\Users\bhupi\Desktop\
11 Dec 2015 10:15:49 AM	11 Dec 2015 09:43:20 AM	11 Dec 2015 09:43:20 AM	11 Dec 2015 09:37:06 AM	4	Fixed	OS	11 Dec 2015 03:53:27 AM	C:\Users\bhupi\Desktop\
11 Dec 2015 10:27:15 AM	11 Dec 2015 09:50:02 AM	11 Dec 2015 09:50:02 AM	11 Dec 2015 09:43:43 AM	6	Fixed	OS	11 Dec 2015 03:53:27 AM	C:\Users\bhupi\Desktop\

Fig. 18. Aggregated data of LNK and DestList stream for each Jump List.

Table 6
Activity timeline of 10 days.

Date	Time	Activity
03 Dec 2015	9:31:57 AM	The system was started
	9:43:07 AM	File#1 was accessed 2*nd* time from *OS* volume
	12:15:28 PM	File#2 was created on *OS* volume
	12:50:20 PM	File#3 was accessed 3*rd* time from *softwares* volume
06 Dec 2015	11:58:42 AM	The system was started
	12:04:04 PM	File#4 was accessed from *softwares* volume
07 Dec 2015	8:58:05 AM	The system was started
	9:13:47 AM	File#5 was created on *OS* volume and accessed
	11:20:49 AM	File#6 was created on removable media named *SANDISK-USB*
	11:22:12 AM	File#6 was accessed again from the same removable media
	11:29:03 AM	File#7 was created on *softwares* volume and accessed
	11:31:04 AM	File#7 was accessed 2*nd* time from the same volume
	11:42:27 AM	File#8 was created on *OS* volume and accessed
10 Dec 2015	9:50:58 AM	The system was started
	10:00:31 AM	File#9 was created on *OS* volume
	10:01:16 AM	File#9 was modified
	10:03:32 AM	File#9 was accessed again from the same volume

destruction or evidence tampering is done then such inconsistencies can be detected by parsing the individual Jump Lists. However, for identification of such deliberate attempts, forensic investigators can combine and correlate the information from other sources such as Windows Shortcut (LNK) files and Prefetch. Forensic investigators can effectively use Jump Lists for gathering information regarding web searches (using Cortana) performed on a suspected system as well as for constructing activity timelines of any user on a suspected system. Also, forensic investigators can use CAT Detect tool developed by Marrington et al. (2011) to detect inconsistencies in the timelines constructed using this type of system logs. The results of the experiments conducted in this paper highlight the added information available in the Jump Lists of Windows 10 and can prove to be promising in the field of forensic investigation.

Further research scope

There are three main areas of research on Jump Lists in Windows 10. The major portions of the DestList stream were successfully identified. However, purpose of few fields remain unknown thus leaving the opportunity to explore more about it. Secondly, carving DestList data of the deleted Entries from a Jump List can be a part of research. Finally, construction of activity timelines requires manual analysis of the extracted data and can be automated in future.

References

Lallie HS, Bains P. An overview of the jumplist configuration file in windows 7. J Digital Forensics Secur Law 2012;7:15–28.

Larson T. Forensic examination of windows 7 jump lists. 2011. http://www.slideshare.net/ctin/windows-7-forensics-jump-listsrv3public [accessed 15.10.2015].

Lyness R. Forensic analysis of windows 7 jump lists. 2012. http://articles.forensicfocus.com/2012/10/30/forensic-analysis-of-windows-7-jump-lists/ [accessed 19.10.2015].

Marrington A, Baggili I, Mohay G, Clark A. Cat detect (computer activity timeline detection): a tool for detecting inconsistency in computer activity timelines. Digit Investig 2011;8:S52–61.

MiTec. Structured storage viewer. 2010. http://www.mitec.cz/ssv.html [accessed 20.11.2015].

MSDN. [ms-cfb]: compound file binary file format. 2015. https://msdn.microsoft.com/en-us/library/dd942138.aspx [accessed 12.11.2015].

MSDN. [ms-shllink]: shell link (.lnk) binary file format. 2015. http://msdn.microsoft.com/enus/library/dd871305(PROT.10).aspx [accessed 11.11.2015].

NirSoft. Jumplistsview. 2013. http://www.nirsoft.net/utils/jump_lists_view.html [accessed 10.09.2015].

Parsonage H. The meaning of linkfiles in forensic examinations [computerforensics.parsonage.co.uk/downloads/themeaningoflife.pdf]. 2010.

Smith GS. Using jump lists to identify fraudulent documents. Digit Investig 2013;9:193–9.

TZWorks. Windows jump list parser (jmp). https://tzworks.net/prototype_page.php?proto_id=20; 2013 [accessed 22.11.2015].

Woan M. Jumplister. 2013. http://www.woanware.co.uk/forensics/jumplister.html [accessed 10.06.2015].