The Forensic Value of the Windows 7 Jump List

Alexander G Barnett

Purdue University

Abstract

The Windows 7 Jump List is an aspect of the Windows 7 operating system that has the potential to contain data and artifacts of great interest to investigators, but has yet to receive any considerable attention or research. As of this writing, only one published work makes mention of their existence, and no tools exist to automate their retrieval and analysis. The goal of this research is to provide an overview of the function and behavior of jump lists, and also to examine the structure of jump lists with the intention of proposing further research for making use of them in a forensic capacity.

Keywords: Jump List, Windows 7, Forensics

Windows Jump Lists have the potential to be an excellent source of evidence for investigators to collect, yet have not been the target of much academic scrutiny. Due to this lack of research, the intent of this paper is provide a basic understanding of the behaviors, data, and structures associated with Jump Lists, and to propose new avenues of research for their exploitation in a forensic capacity. The first section of this paper will provide an overview of Jump Lists and their end-user functions, while the second will document several experiments to determine how Jump Lists behave under certain circumstances. The third section will examine the internal structure of the Jump List, and the fourth and final section will recommend future research.

Overview

Jump Lists are a feature new to Windows 7. They serve a number of purposes, depending on the specific program utilizing the jump list. For example, the Microsoft Word Jump List contains a list of recently opened documents, as well as a section for user-defined "pinned" documents that never leave the list.

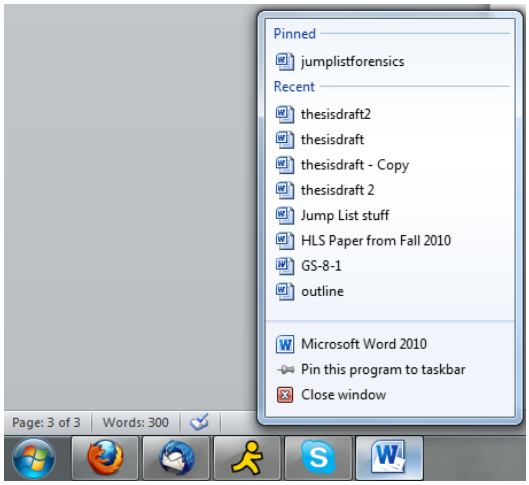

Figure 1. The Microsoft Word taskbar Jump List.

Jump Lists may also appear in the Start Menu, where they duplicate the functionality of taskbar Jump Lists.

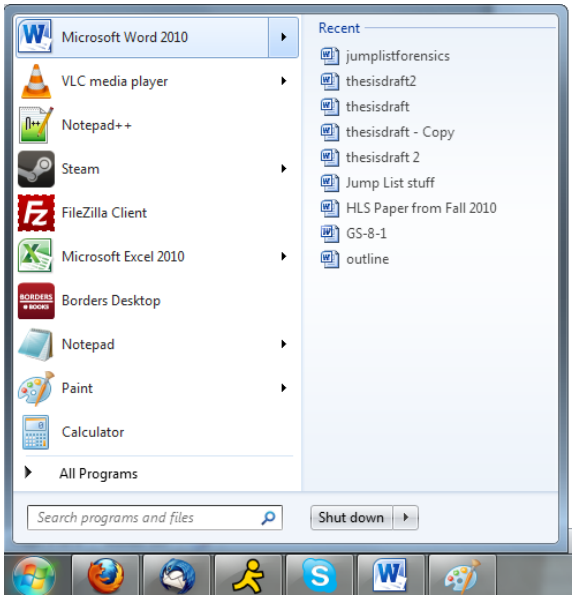

Figure 2. The Microsoft Word Jump List in the Start Menu.

The Microsoft Word Jump List demonstrates the basic function of Jump Lists, which is primarily to show files recently used by specific programs. Other programs, such as AOL Instant Messenger, utilize their Jump Lists for additional tasks.

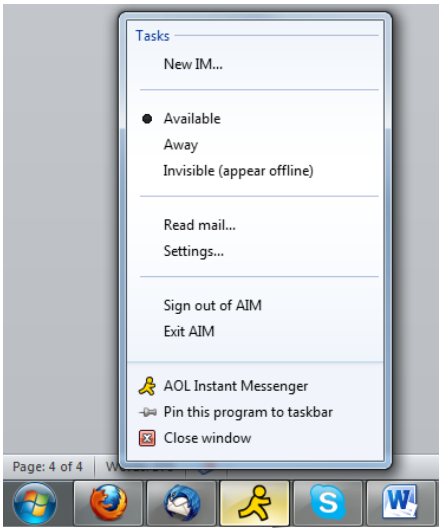

Figure 3. The AOL Instant Messenger Jump List.

AIM's Jump List contains shortcuts for sending new instant messages, changing the user's online status, reading their mail, and changing the program's settings. While program shortcuts may be useful, it is the recent items feature that is of primary interest to investigators. These lists may contain files that have been created, downloaded, uploaded, or opened, depending on the program being used, and retain the items even after the actual file is deleted. What's more, the Jump List may record items the user does not know are being recorded (such as files downloaded while in Firefox private browsing mode), making them even more useful for investigators.

Completely clearing a single Jump List is not an intuitive process. There are three methods the user can utilize: The first is to manually remove each item in the list by right clicking on the Jump List entry and selecting "Remove from this list", which is not practical for larger lists. The second method is to right click on the Start Menu icon and select Properties, select the Start Menu tab, and uncheck both options in the Privacy section, which clears all Jump Lists, not just one. The third method is to manually delete the Jump List file located at:

%APPDATA%\Microsoft\Windows\Recent\AutomaticDestinations or

%APPDATA%\Microsoft\Windows\Recent\CustomDestinations.

This last item is of particular interest for a number of reasons. First, the *AutomaticDestinations* and *CustomDestinations* folders are hidden. This would normally not be an issue to an experienced user, however even after changing the Windows settings to show hidden files and folders, these particular folders remain hidden. This unusual behavior means that, unless the exact path of the folder is known, the user cannot access individual Jump List files. Additionally, even if the user were able to locate the Jump List files, deleting the correct

list would be difficult given that Jump List file names are encoded in a seemingly random string of characters. Without viewing the contents of the file in a HEX editor as depicted in Appendix A, it is impossible to know which program the list represents. Due to the complexities involved with deleting Jump List data, the records they contain represent a good source of possible evidence regarding user activities for investigators.

Experiments

The following experiments were carried out on a PC running Windows 7 Ultimate (64-bit). The Jump List files were manually deleted from the *AutomaticDestinations* and *CustomDestinations* directories at the beginning and conclusion of each experiment. Experiments are grouped by the program being tested.

Firefox 3.6.16

Experiment #1: Downloading images in normal browsing mode.

For this experiment, images were downloaded from various websites by right-clicking on the image selecting “save-as”.

Result: Images saved in this way appeared in the Firefox Jump List after downloading completed.

Experiment #2: Downloading images in private browsing mode.

This experiment mimics experiment #1, with the exception of placing the browser in private browsing mode before beginning.

Result: Images saved appeared in the Firefox Jump List and remained there after the program was closed. These images did not appear in the Firefox disk cache, which was viewed through the about:cache Firefox interface.

Experiment #3: Uploading images in normal browsing mode.

For this experiment, images were uploaded to an online image board through the site's upload function.

Result: Uploaded images were listed in the Firefox Jump List. Interestingly, the images appeared in the Jump List immediately upon being confirmed in the file dialog box, not after being uploaded to the server.

Experiment #4: Uploading images in private browsing mode.

For this experiment, images were uploaded to an online image board through the site's upload function while Firefox was in private browsing mode.

Result: Uploaded images were listed in the Firefox Jump List. Again, the images appeared in the Jump List immediately upon being confirmed in the file dialog box, not after being uploaded to the server.

Experiment #5: Comparing the Jump Lists of uploaded and downloaded files.

For this experiment, an image was uploaded to an image board in normal browsing mode. After uploading, the Jump List file was copied to another folder and renamed upload.automaticDestinations-ms. After this, the original Jump List file was deleted and the same image downloaded from the image board. The new Jump List file was copied to another

folder and renamed download.automaticDestinations-ms. The MD5 hash value of each file was then calculated.

Result: The hashes did not match. Viewing the HEX values of each file revealed substantial differences. This could indicate that Jump Lists have an internal mechanism for differentiating uploaded and downloaded files.

Experiment #6: Uploading a file to a flash-based website.

For this experiment, an image file was uploaded to a flash-based website through Firefox in normal browsing mode. The upload function also appeared to be flash-based, although it used a standard file browser dialog.

Result: The file did not appear in the Firefox Jump List.

Table 1. Summary of Firefox Results.

Experiment	Result
1. Downloading an image in normal browsing mode	Item appeared in the Jump List.
2. Downloading an image in private browsing mode.	Item appeared in the Jump List.
3. Uploading an image in normal browsing mode.	Item appeared in the Jump List.
4. Uploading an image in private browsing mode.	Item appeared in the Jump List
5. Comparing Jump Lists containing one downloaded file and one uploaded file.	MD5 hash values did not match.
6. Uploading a file to a flash-based website.	Item did not appear in the Jump List.

Firefox – Conclusions

The most notable aspect of the Firefox Jump List's behavior is that files downloaded and uploaded, even in private browsing mode, are recorded. If the user overlooks this behavior, the Firefox Jump List could provide a telling log of activities performed online. Additionally, the fact that a Jump List containing only one item downloaded and a Jump list containing only one item uploaded are different reveals that Jump Lists may have some sort of mechanism for differentiating how the file was processed through the browser. This feature could become vitally important if, for example, a case wanted to prove that a user distributed a file rather than merely acquired it. Finally, the fact that an image uploaded through a flash-based interface does not appear in the Jump List reveals that Jump Lists do not record 100% of file uploads, and may in fact omit downloads and uploads from other methods as well.

Internet Explorer 8

The experiments performed on the Firefox browser were performed on Internet Explorer. For the sake of brevity, the procedures will not be re-listed.

Experiment #1: Downloading images in normal browsing mode.

Result: The file did not appear in the Internet Explorer Jump List in the Windows UI. However, viewing the *AutomaticDestinations* Jump List's HEX data revealed that the image was recorded.

Result 2: This test was repeated on a later date. A file was saved immediately after deleting the Jump List, after which the file was listed in the Jump List UI. However, after

visiting several websites, the file disappeared from the list and was replaced by links to the websites recently visited.

Experiment #2: Downloading images in private browsing mode.

Result: The file did not appear in the Internet Explorer Jump List in the Windows UI. Also, it did not appear in the Jump List HEX data.

Experiment #3: Uploading images in normal browsing mode.

Result: The file did not appear in the Internet Explorer Jump List. However, viewing the *AutomaticDestinations* Jump List's HEX data revealed that the image was recorded.

Experiment #4: Uploading images in private browsing mode.

Result: The file did not appear in the Internet Explorer Jump List. Also, it did not appear in the Jump List HEX data.

Expirement #5: Comparing the Jump Lists of uploaded and downloaded files.

Result: The hashes did not match. Viewing the HEX values of each file revealed substantial differences.

Table 2. Summary of Internet Explorer Results.

Experiment	Result
1. Downloading an image in normal browsing mode	Item did not appear in Jump List UI, however it appeared in the Jump List HEX values.
2. Downloading an image in private browsing mode.	Item did not appear in the Jump List.
3. Uploading an image in normal browsing mode.	Item did not appear in Jump List UI, however it appeared in the Jump List HEX values.
4. Uploading an image in private browsing mode.	Item did not appear in the Jump List.
5. Comparing Jump Lists containing one downloaded file and one uploaded file.	MD5 hash values did not match.

Internet Explorer – Conclusions

During the course of these experiments, it was noted that after visiting the same page several times, a link to the page was stored in the Internet Explorer Jump List. Also, these entries persisted after deleting the files located in the *AutomaticDestinations* directory, indicating the existence of a separate Jump List file. After researching the topic online, it was discovered that another set of Jump List files do indeed exist at *%APPDATA%\Microsoft\Windows\Recent\CustomDestinations*. Purging these files removed the entries in the Internet Explorer Jump List UI, showing that a second Internet Explorer Jump List is stored in this location.

This discovery prompted the researcher to revisit experiments #2 and #4 (downloading and uploading images in private browsing mode) to see if the files were noted in the *CustomDestinations* Jump List. After re-performing the tests, it was noted that this second Jump List made no mention of the files either.

It was also interesting to note that the files uploaded and downloaded in private browsing mode were not stored in the Internet Explorer Jump List, while files uploaded and downloaded in

private browsing mode using the Firefox browser were. This difference may indicate that Internet Explorer, being a core part of the operating system, has access to system methods inaccessible to other browsers.

It was noted that even after clearing both sets of Jump List files, upon restarting the browser, the Jump List referencing frequently visited sites was restored. However, after re-deleting the Jump List files, clearing the browser history, and restarting the browser, the Jump List was not repopulated. Based on this observation, it is reasonable to assume that this particular Jump List can be automatically generated from the browser's history files if deleted.

Finally, it was discovered through Experiment 1, Result 2 that the Internet Explorer Jump List will default to the *AutomaticDestinations* list if the *CustomDestinations* list is not available. However, it will revert back to the *CustomDestinations* list as soon as it is available.

Jump List File Names

Jump List file names, while appearing to be a random string of characters, always follow the format *16 characters dot automaticDestinations-ms* or *customDestinations-ms* (depending on which folder the file is present in). The 16 characters preceding the *.automaticDestinations-ms* or *.customDestinations-ms* are of particular interest to investigators, since they identify which program the list is associated with. The following tests will attempt to shed some light on how these lists are named.

Experiment #1: Are Jump List names static or dynamic?

The 16 characters appear to be random. In this experiment, several different programs' Jump Lists will be deleted and recreated multiple times to determine if the name is randomly generated.

Result: The file names did not change. Additionally, it was found that if a program had Jump Lists in both the *AutomaticDestinations* and *CustomDestinations* folders, the 16 character identifier was the same on both files. Table 3 lists the Jump List names of several programs:

Table 3. Jump List names.

Program Name	Jump List Name
Firefox 3.6.16	5c450709f7ae4396
Internet Explorer 8	28c8b86deab549a1
Microsoft Word 2010	a7bd71699cd38d1c
Windows Explorer	1b4dd67f29cb1962
Notepad (64-bit)	9b9cdc69c1c24e2b
Notepad (32-bit)	918e0ecb43d17e23

File Names – Conclusions

The fact that the identifiers are static and always sixteen characters long reveals that they are most likely encoded names of whichever program they represent. However, it is not clear at this time how these characters are encoded. Although they appear to be hexadecimal representations of characters (given that the characters stop at the letter f), translating the strings from HEX to ASCII text produces no meaningful results. It is possible that decoding the identifier would produce an eight-character DOS name, although this raises the question of how

Windows can differentiate between programs with identical names (such as the 32 and 64 bit versions of Notepad).

File Structure

Examining the file structure of a Jump List is a difficult task. Viewing the file in plain text produces garbage text, so the only available method is to view the contents in a HEX editor. Even after viewing the file in HEX, making sense of the ensuing code is quite difficult. However, close examination reveals some commonalities in the structure of each Jump List. To begin, all Jump List files appear to begin with code depicted in figure 4:

```
00000000 d0 cf 11 e0 a1 b1 1a e1 00 00 00 00 00 00 00 00 ÐÏ.à¡±.á........
00000010 00 00 00 00 00 00 00 00 3e 00 03 00 fe ff 09 00 ........>...þÿ..
00000020 06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 ................
00000030 01 00 00 00 00 00 00 00 00 10 00 00 02 00 00 00 ................
```

Figure 4. The first four lines of Jump List code.

The next line is nearly identical in all files, with the exception of the first value which is always either “01” or “02”. It is unknown at this time what this value signifies.

```
00000040 01 00 00 00 fe ff ff ff 00 00 00 00 00 00 00 00 ....þÿÿÿ........
```

Figure 5. The fifth line of code (Notepad 64 bit)

```
00000040 02 00 00 00 fe ff ff ff 00 00 00 00 00 00 00 00 ....þÿÿÿ........
```

Figure 6. The fifth line of code (Firefox)

The next 27 lines contain the code shown in figure 7, after which the code is no longer uniform from list to list.

```
00000050 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff yyyyyyyyyyyyyyyy
```

Figure 7. Lines six through thirty-two.

The ensuing structure is complex and seemingly repetitive. The complete file can be roughly represented as having the structure depicted in table 4. Figure 8 shows a partial view of the Firefox Jump List after uploading a file named cat.JPG to a website. There are several observations worth noting in the example. First, while the file path is shown three times in this particular figure, partial file paths are listed multiple times in the preceding code with large sections of unknown code in between, as shown in Figure 9. Also worth noting is the second file path, which contains the machine's host name (MAGUS in this case). Downloading a file produces a similar structure, with multiple repetitions of the file path along with host name. Comparing the structures of a Jump List with one uploaded file and a Jump List with one downloaded file reveals significant differences between the two files' structures, however there are no discernable sections of code which clearly indicate whether a file was uploaded or downloaded. Both lists contain references to the hostname and users@Shell32.dll, and both contain both full and partial file paths repeated many times throughout the file, although in seemingly different orders with small sections of illegible code interspaced. However, one immediately noticeable difference is that the Jump List containing the downloaded file is much longer (by approximately 2600 characters) than the Jump List containing the uploaded file. Although much of the data contained in a Jump List is incomprehensible at this time, investigators can still make use of the file paths they contain to prove that a file was used on that particular machine.

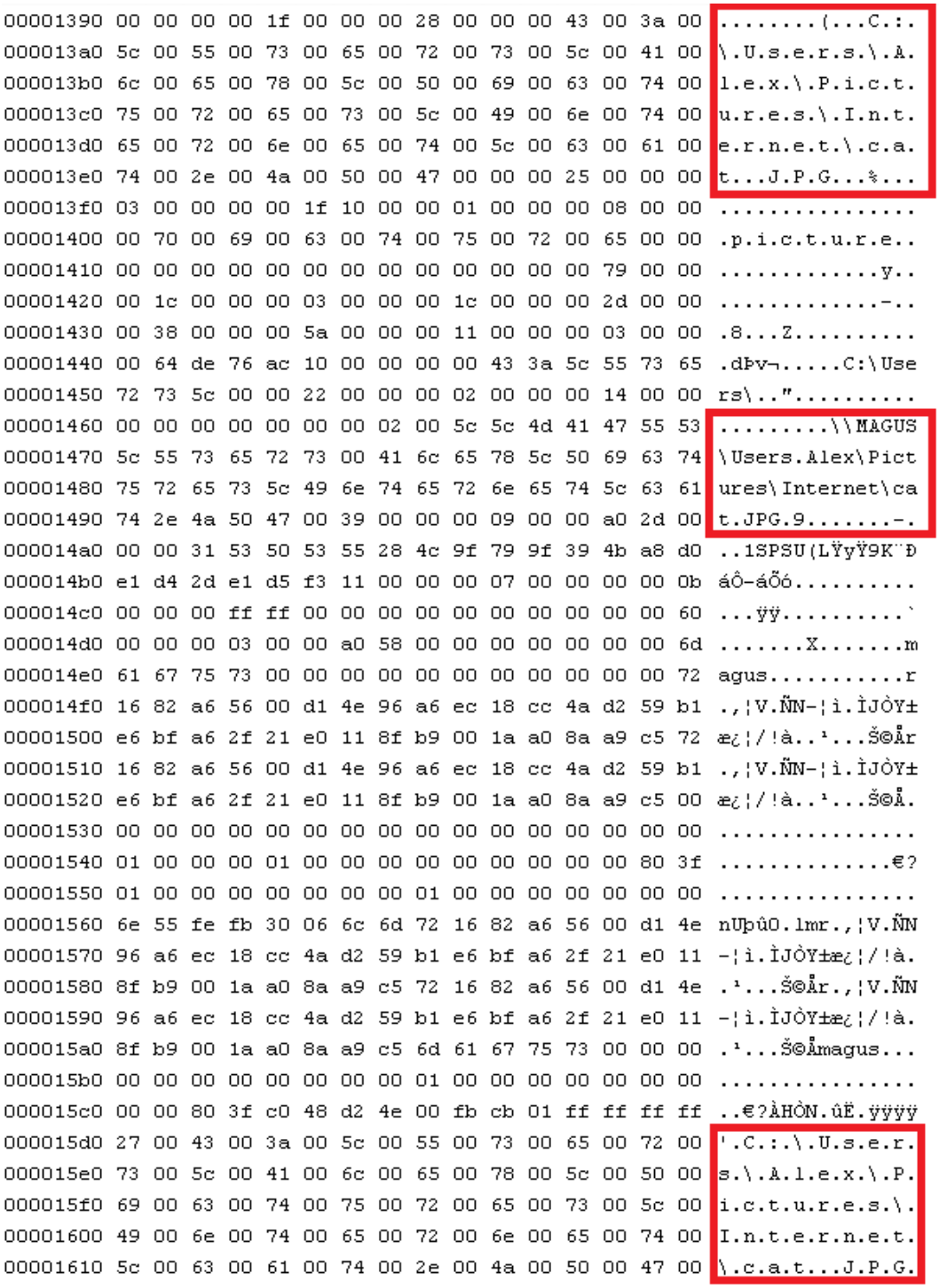

Figure 8. Section of Firefox Jump List after uploading one file.

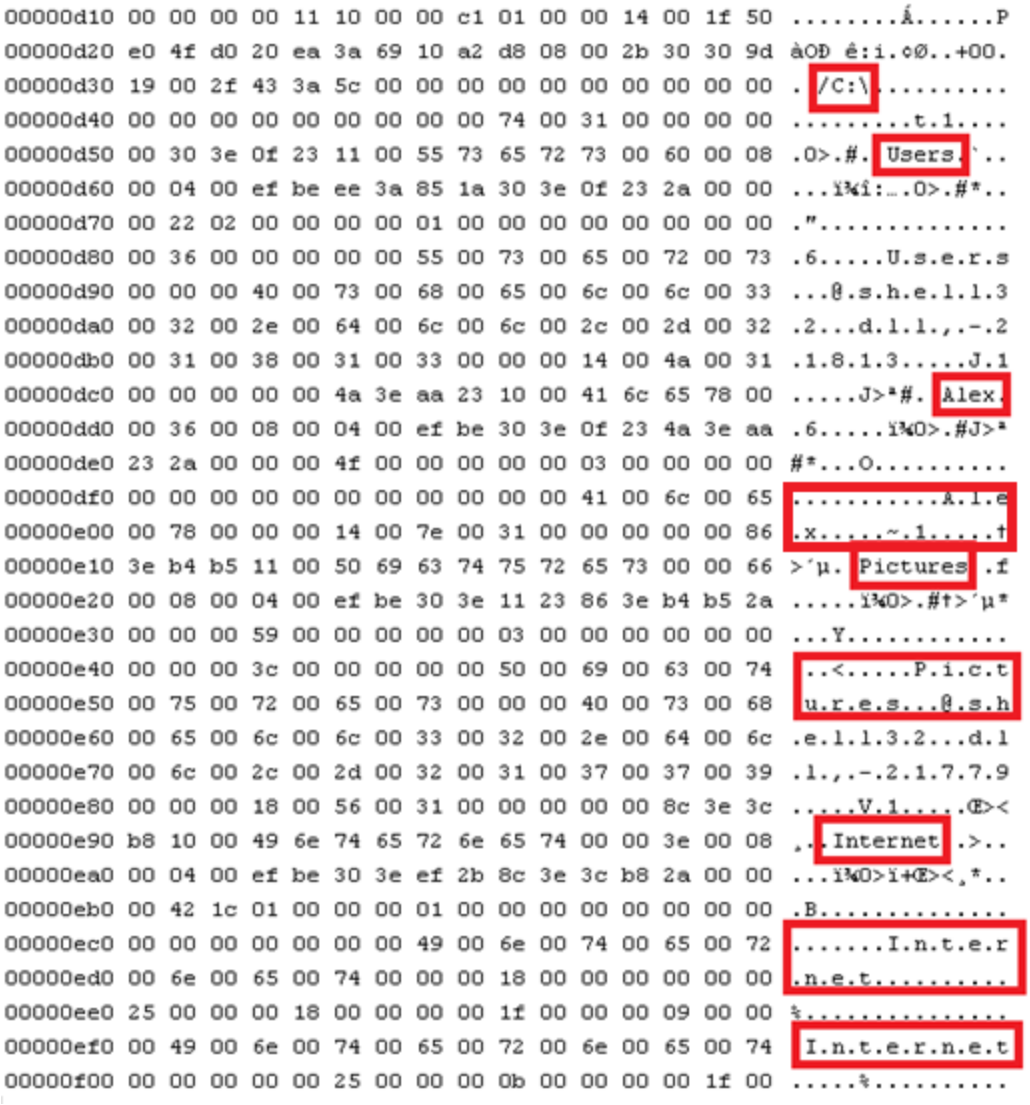

Figure 9. Path fragments in the Jump List.

Table 4. Rough Jump List file structure.

Header
Padding
File path fragments
Full file paths
Padding
File path fragments
Full file paths

Conclusions and Recommendations

Jumps Lists have a number of practical applications for investigators. At minimum, they provide a list of files uploaded, downloaded, viewed, created, or otherwise used by every program on the system. Additionally, they can also serve as a log of frequent tasks undertaken by the user with some programs, such as with sites frequently visited on Internet Explorer. What's more, with further research it may be possible to determine how the file was processed through the program it is listed in, such as uploaded vs. downloaded, or in the case of a word processor, created vs. opened. In terms of practical application, this could mean proving an illegal image was uploaded to a server rather than downloaded, or that a ransom note was written and saved rather than merely opened. Any situation that needs to prove that a file was used on a system could potentially benefit from Jump List data.

Future research will ideally lead to the development of a tool that can automatically process Jump Lists as evidence. To reach this point, a number of research goals must first be

accomplished. First, deciphering the 16 character identifier in the Jump List name will allow the identification and classification of Jump List files. Failing that, a comprehensive list of programs and their associated identifiers could be developed, however this is not ideal. Second, a greater understanding of how Jump Lists operate must be attained. The internal structure of the list must be deciphered to identify what information is actually stored by the list, other than the file name, host name, and path to the file. Finally, a program must be developed to automatically identify a list, extract the file paths, and tag each file with whatever attributes can be identified (uploaded, downloaded, etc). With the completion of all these tasks, Jump Lists will be of great use to investigators.

APPENDIX A – DETERMINING JUMP LIST ASSOCIATION

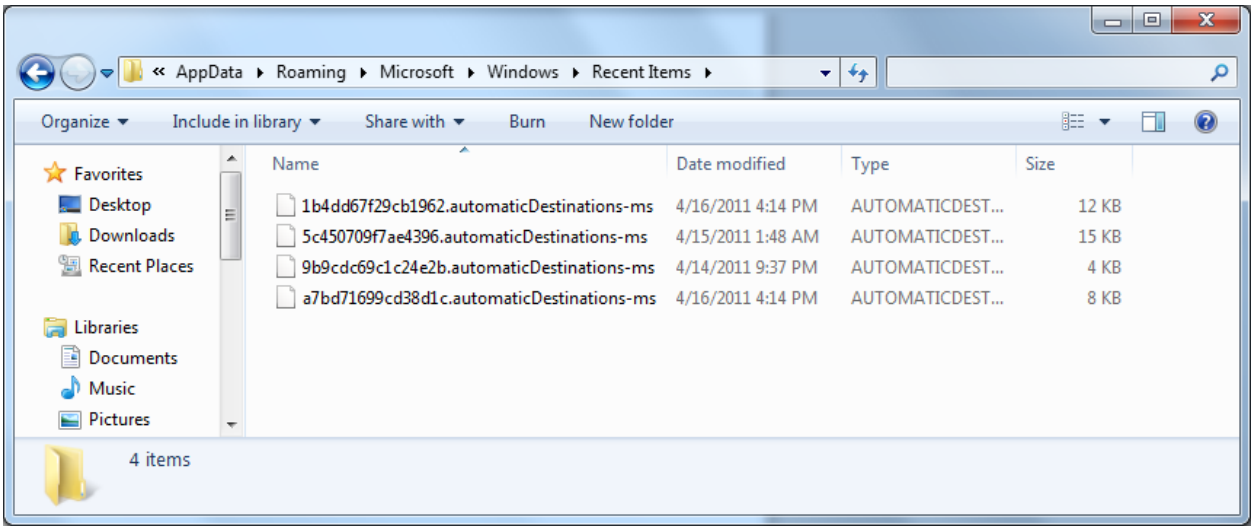

Figure A.1. The *AutomaticDestinations* Folder.

To determine which file belongs to which program, first view the selected program's Jump List in the Windows UI. Firefox was chosen for this example, as shown in figure A.2.

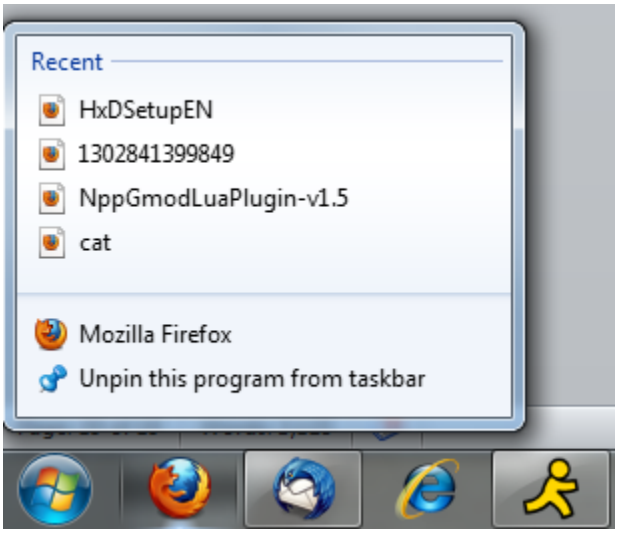

Figure A.2. Firefox Jump List.

From here, open each Jump List file in a HEX editor and search for the entries present in the list. By this method, we can determine which file belongs to which program.

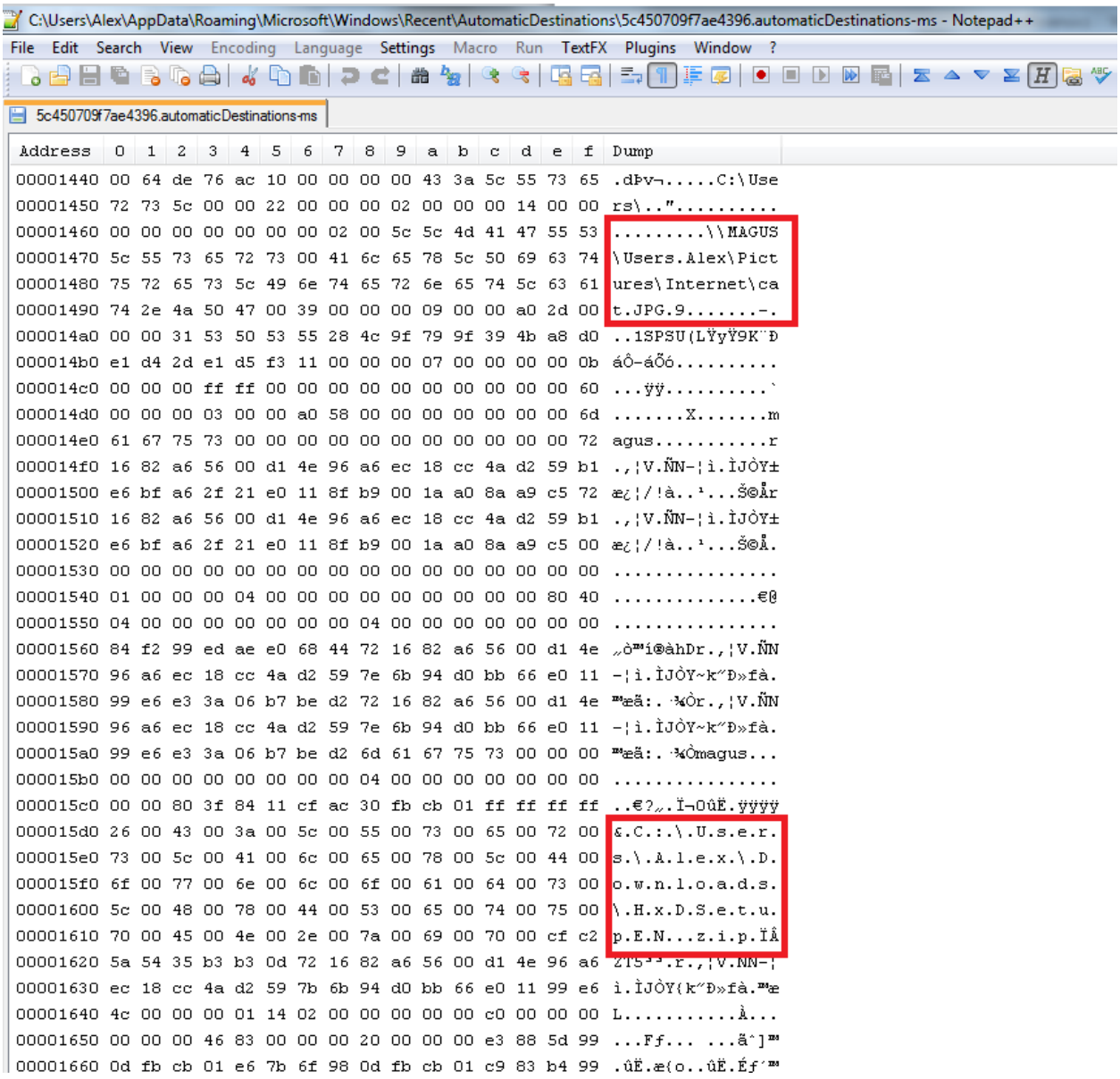

Figure A.3. File entries in the Firefox Jump List.