Windows File Analyser Guidance—Allan S Hay

** The following information is a guide to understanding the Prefetch Folder and Windows Shortcut File Format (LNK) and all work undertaken in my research, should be verified by the analyst. Prior to release this product was tested in the Forensic Computing community, and some of their findings are incorporated into this guidance. All output from WFA should be verified by the analyst. **

Allan S Hay

November 2005

The program is divided into 3 parts, which the latter 2 I go into some depth to explain the workings of how the data is derived.

Thumbnail Analyser

This program in most circumstances, will extract the OLE embedded data from a Thumbs.db file and present the information in a visible format. The larger the size of file being examined, directly affects the time for the Objects to be rendered.

Although it may look that the file is not being processed, be patient. Select a single image (Save) to be exported, or select (Report). The number on the top left of the report (Count) is the number of images rendered.

For a thorough explanation of the Forensic value of the Thumbs.db structure an excellent source of information can be found at:

http://www.accessdata.com/files/whitepapers/tdb.pdf

Prefetch Analyser

The Prefetch cache is resident on a Windows XP OS. Microsoft created a Prefetch cache to improve boot and application launch time. The functionality of how the cache performs is based on a Registry key which can be found here (HKEY_LOCAL_MACHINE\ SYSTEM\ CurrentControlSet\ Control\ Session Manager\Memory Management\PrefetchParameter).

The Prefetch cache purpose is to speed up processes at boot and during application run time. By cachingcommonly used applications the OS can determine to apportion system resources in anticipation that the user will access the application. When an application is launched the system updates an entry in the path **C:/Windows/Prefetch** with the name of the application and a file extension (.pf). The file contains among other items the last time that the file was modified as a 64bit HEX value time, and increments an integer on how many times the application has been run.

Before I explain where the information is, please bear in mind that the Prefetch cache is taking information from the boot process and can take information from Scheduled Tasks. Therefore, it would be prudent to access the RUN (NTUSER/Software/Microsoft/Windows/Explorer/RUN) key in the registry and also the TASKS (C:/Windows/Tasks). After this scour the drive to determine if there is any 3rd party software scheduling resident on the drive.

Preftech Folder

It would appear that in system idle time every 3-5 days (but can be longer), depending on how the caching is determined, the folder will purge itself. The analyst can search across the drive using a GREP expression and pull out the data from the Unallocated Clusters. By scrutinising the data structure the analyst should be able to determine the data from the two forgoing offsets, minus temporal data.

The following should be used to search across the Unallocated Clusters to determine the historic Prefetch Files.

(….SCCA.)

or:

("\\x11\\x00\\x00\\x00\\x53\\x43\\x43\\x41\\x0F", FileClass::GREP);

Using a hex editor program, open a file in the Prefetch folder, try finding an .exe that you would use regularly, perhaps MS Word. In your hex editor, go to File Offset 120. You will see an 64 bit hex value, which is the date that the executable was last summoned, and this gets updated in the PF file. Go to File Offset 144, note the value. This is the number of times that the application has been run, since the creation of the .pf file.

This is an OS dependent folder, so the examiner has to understand that PFA will only work on Windows XP. Hopefully Windows Vista will include a Prefetch or equivalent. Before using PFA, it is best to see where the data you are interpreting is located, and using a hex editor you will be able to see the changes being made. MiTec HexEdit can be downloaded from the link below:

http://www.mitec.cz/Downloads/HEXEdit.zip

The research work that I have carried out has been evolved into Version 1.0 of the Prefetch Folder Analyser, which Michal Mutl has coded for me. The program will automatically read the local drive of the user when first initialized. Within the main viewing pane is the breakdown of the resident data. Navigate with Hexedit to your Prefetch Folder on your OS and select a file which is in regular use. Open a .pf file in HexEdit and you should see an image similar to the one on the following page.

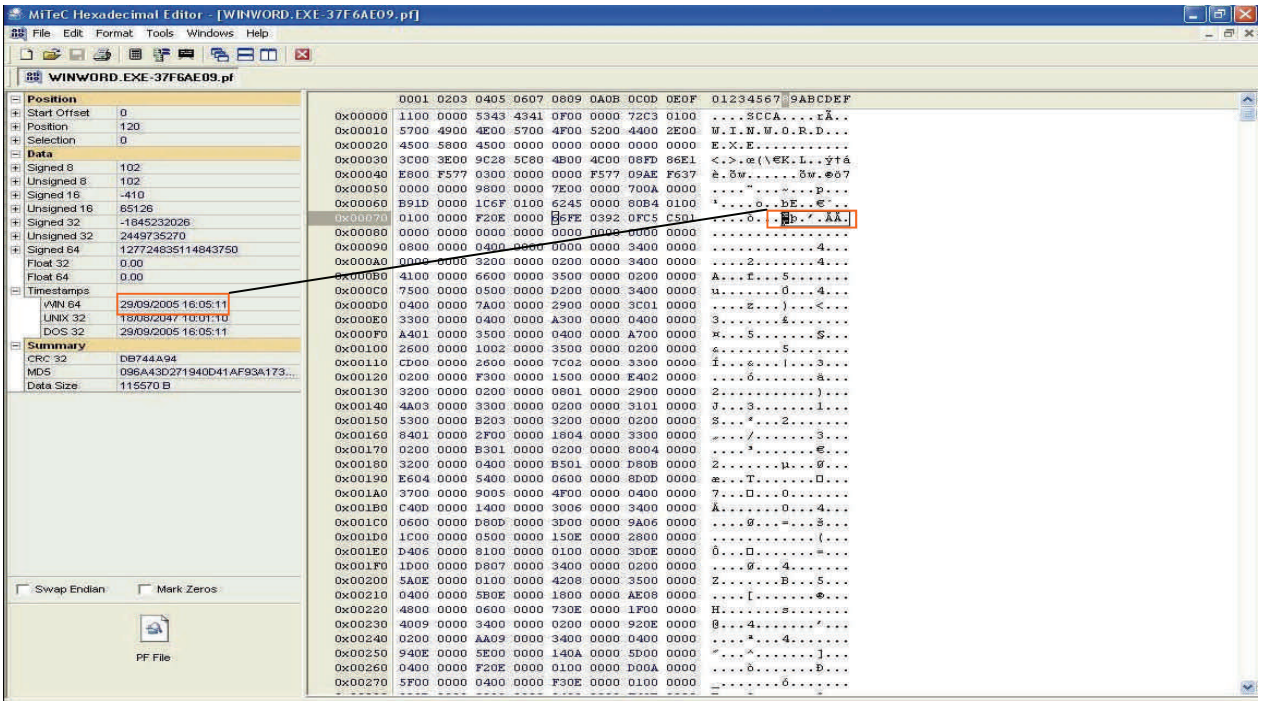
Figure 1.0

The following is an image captured from the Winword.pf when viewed via Hexedit. Note the File Offset 120 which is a 64 bit Windows Time entry. By placing the cursor on the first byte, the Hexedit program enumerates the string and this can be seen on the left hand panel. Note this time for later.

Prefetch Folder

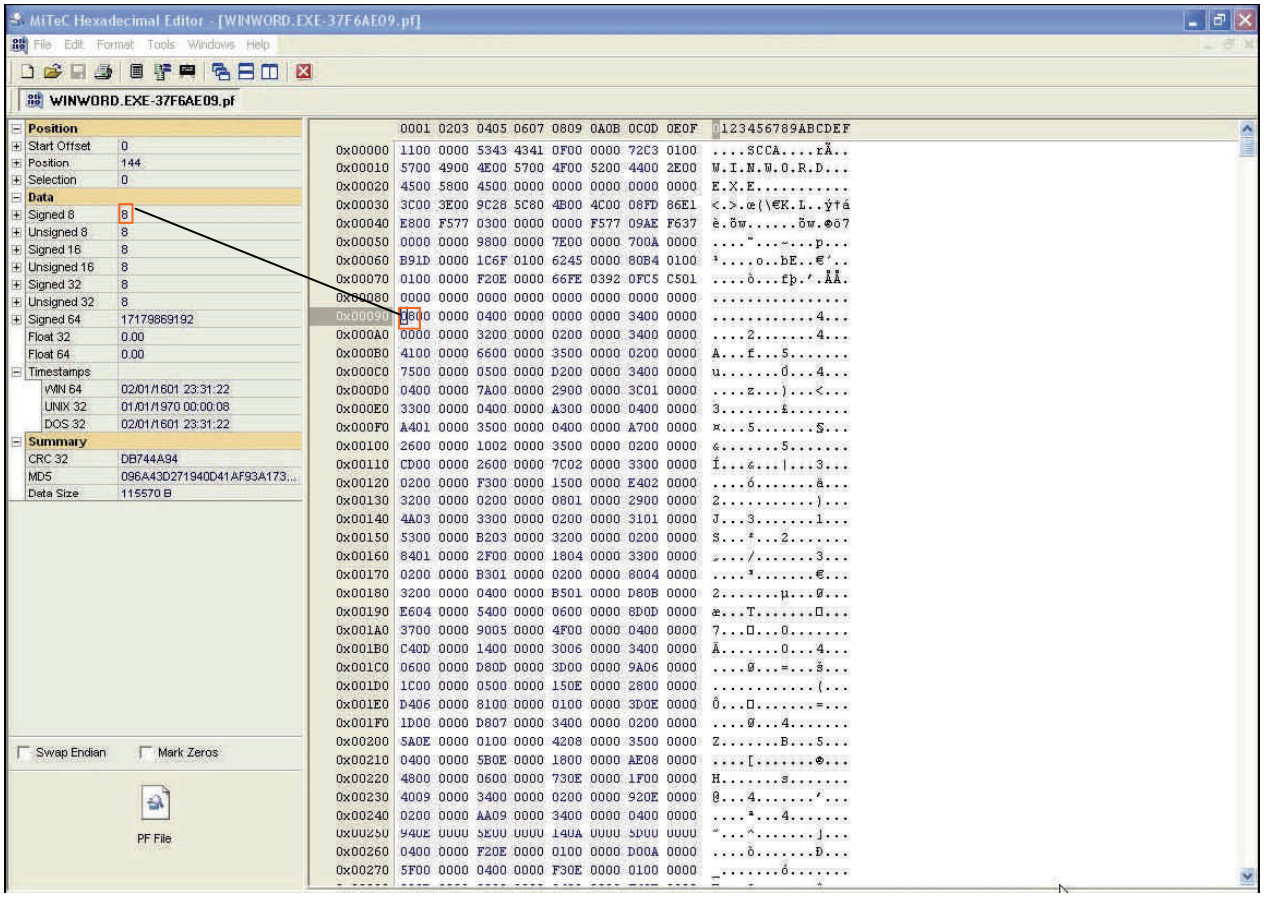
Figure 1.0

The following is an image captured from the Winword.pf when viewed via Hexedit. Note the File Offset 120 which is a 64 bit Windows Time entry. By placing the cursor on the first byte, the Hexedit program enumerates the string and this can be seen on the left hand panel. Note this time for later.



This image is the same as the previous, but File Offset 144 is highlighted which is the incremental integer. The value interpreted with Hexedit can be seen in the left hand panel in the Data section.

Figure 1.1

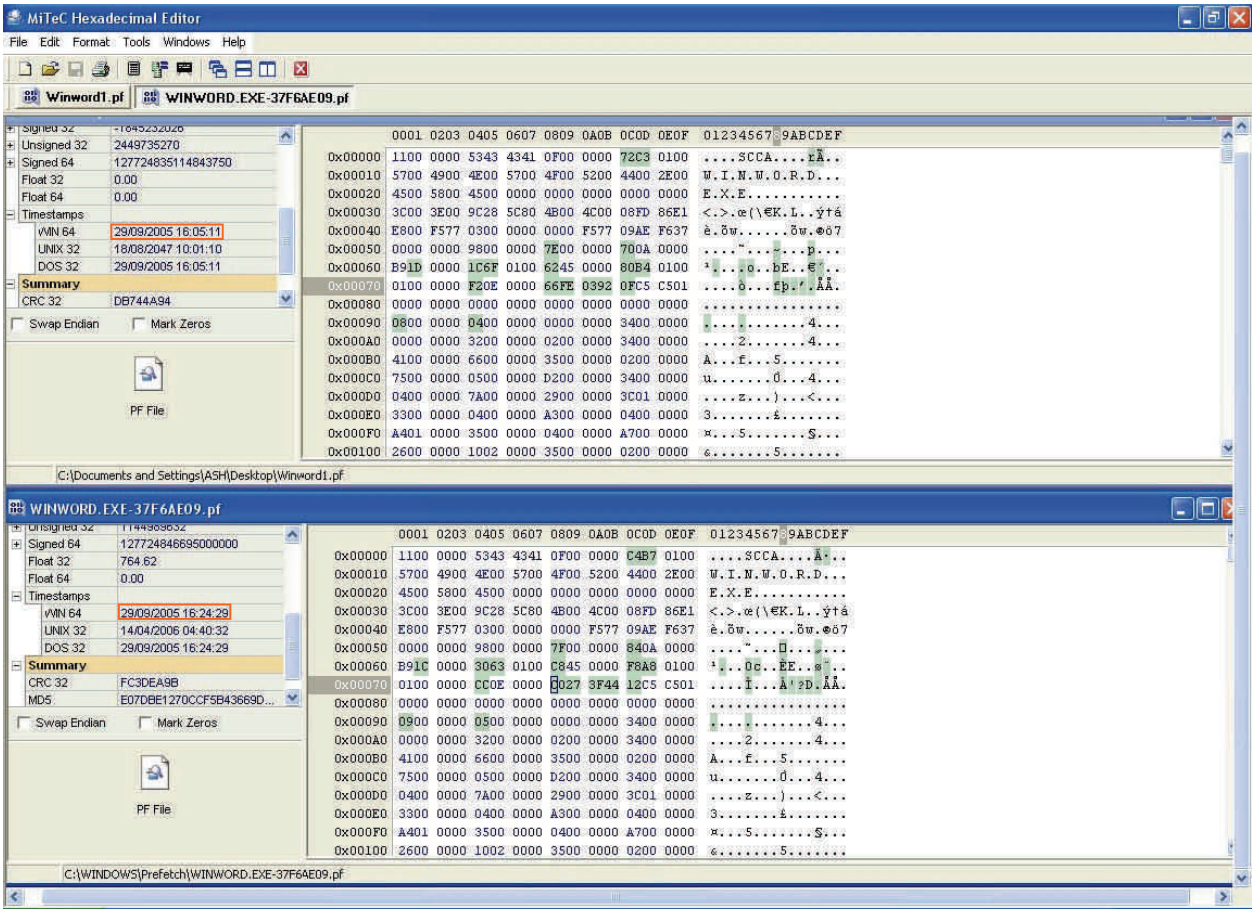


Prefetch Folder

As can be seen from the header information I have accessed a .pf file for WinWord (MS Word).
I then run MS Word, only the application is needed, not a proprietary file type associated with MS Word. I do not have to open a document, only the application needs to be summoned.

Using Hexedit it is possible to differentiate between 2 files, when placed within Hexedit for comparison.
I loaded the original Winword.pf and a previously saved Winword .pf file prior to using MS Word.
It is possible to see the data that has changed. In this case look at the two 64 bit File Time differences.
Both are highlighted in the panel on the left.

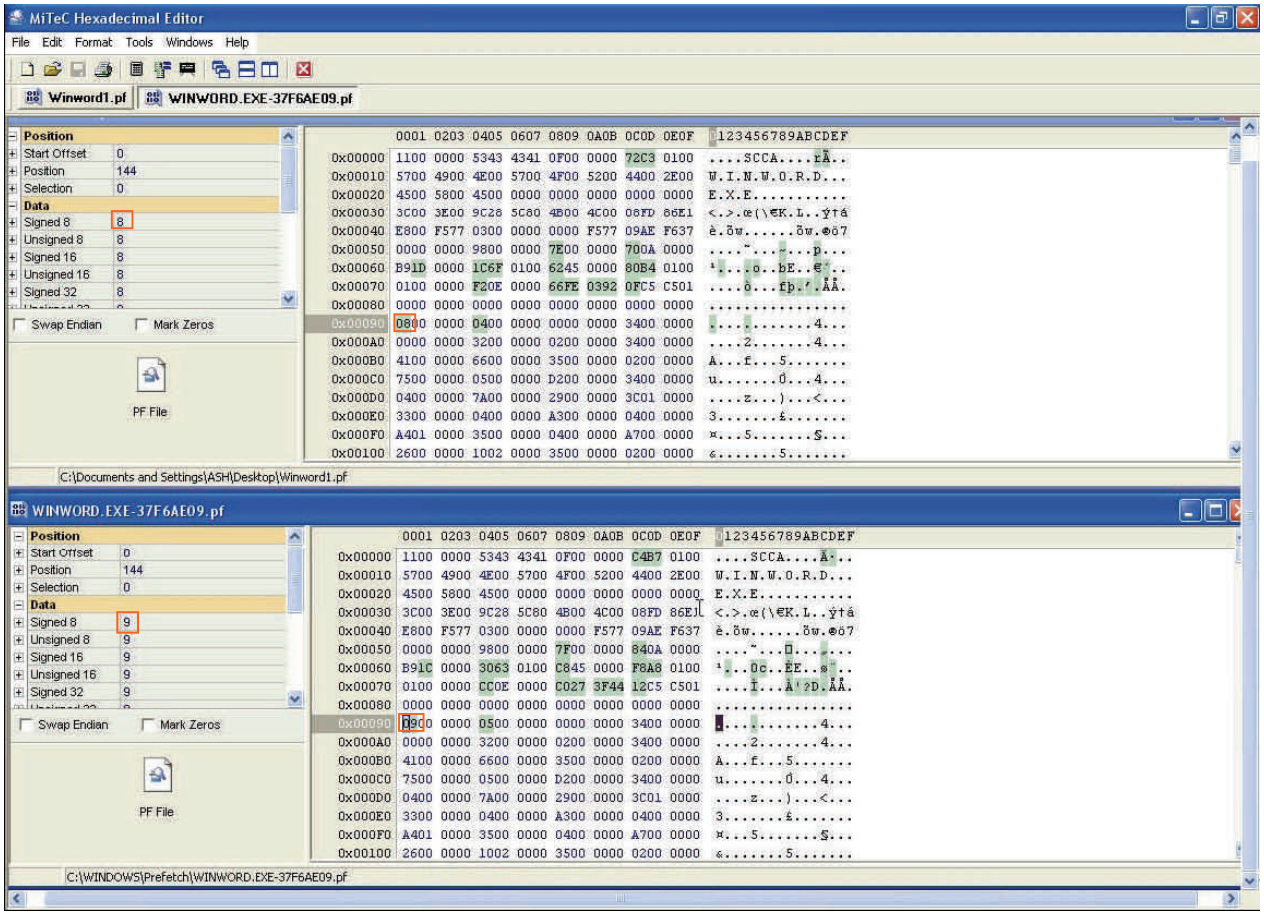
Figure 1.2



For your information when using MiTec HexEdit. When the user compares two files, the area shaded in green represents the comparative differences between the two files.

The last image in the section shows the differences as at File Offset 144. In Figure 1.0, the integer was showing a value of 8 runs. Note that I used MS Word on only one occasion and the value has incremented by one.

Prefetch Folder



Using PFA

The program has been developed to output the following information which may be of interest of the analyst.

Application:
This column highlights the name of the application or program. In research work, I found that some programs that are not installed, that is stand alone, will not have any entries in the Prefetch Folder. Or entries that are standalone can have erroneous runs.

Created Date:
When the system flushes the cache as and when it sees fit, (MS state 3 days, though I do not agree with this), a new .pf file should be generated when the program is either executed, or on next boot up, a new .pf file will created. The first time that the program is used the .pf will increment by one and so on. If the .pf file is not flushed, then from time the .pf file being created, along with the value at File Offset 144, should give an indication of how many times that the program has been used.

Written:
Easily understood as the time and date at which the file was written to.

Last Accessed:

This is not to be interpreted as the last time that the file was accessed to update, this could have been accessed by a third party utility.

Prefetch Folder

Embedded Date:

This date as at File Offset 120, should marry with Written Date. The reason that the Embedded Date was extracted is due to the fact that when these files become purged, it is possible to search in the UC for these files given the header information. I have run a bastardized EnScript to pull out the information and I will make this available ASAP.

Runs:

The value/s returned at File Offset 144.

File Path Hash:

See Appended**.**

MD5:

A hash value calculated on the unique data embedded with each .pf file.

Practical Use.

When 'Analyse Prefetch' is selected it will automatically check for the default location of the Prefetch folder, normally C:/Windows/Prefetch.

The user should now see the entries of their local Prefetch Folder. Expand the Window to view all the detail.

To view a foreign Prefetch Folder, the volume must be mounted in either VM or proprietary product like Mount Image Pro, in order to preserve the MAC times.

Appendix

The information below was drafted from the MSDN site and was accurate at the date of publishing.

http://msdn.microsoft.com/msdnmag/issues/01/12/XPKernel/default.aspx

All versions of Windows except real-mode Windows 3x are demand-paged operating systems, where file data and code is faulted into memory from disk as an application attempts to access it. Data and code is faulted in page-granular chunks where a page's size is dictated by the CPU's memory management hardware. A page is 4KB on the *x*86. Prefetching is the process of bringing data and code pages into memory from disk before it's demanded.

In order to know what it should Prefetch, the Windows XP Cache Manager monitors the page faults, both those that require that data be read from disk (hard faults) and those that simply require that data already in memory be added to a process's working set (soft faults), that occur during the boot process and application startup. By default it traces through the first two minutes of the boot process, 60 seconds following the time when all Win32 services have finished initializing, or 30 seconds following the start of the user's shell (typically Microsoft Internet Explorer), whichever of these three events occurs first. The Cache Manager also monitors the first 10 seconds of application startup. After collecting a trace that's organized into faults taken on the NTFS Master File Table (MFT) metadata file (if the application accesses files or directories on NTFS volumes), the files referenced, and the directories referenced, it notifies the Prefetch component of the Task Scheduler by signalling a named event object.

Prefetch Folder

In order to know what it should Prefetch, the Windows XP Cache Manager monitors the page faults, both those that require that data be read from disk (hard faults) and those that simply require that data already in memory be added to a process's working set (soft faults), that occur during the boot process and application startup. By default it traces through the first two minutes of the boot process, 60 seconds following the time when all Win32 services have finished initializing, or 30 seconds following the start of the user's shell (typically Microsoft Internet Explorer), whichever of these three events occurs first. The Cache Manager also monitors the first 10 seconds of application startup. After collecting a trace that's organized into faults taken on the NTFS Master File Table (MFT) metadata file (if the application accesses files or directories on NTFS volumes), the files referenced, and the directories referenced, it notifies the Prefetch component of the Task Scheduler by signaling a named event object.

The Task Scheduler then performs a call to the internal NtQuerySystemInformation system call requesting the trace data. After performing post-processing on the trace data, the Task Scheduler writes it out to a file in the \Windows\Prefetch folder. The file's name is the name of the application to which the trace applies followed by a dash and the hexadecimal representation of a hash of the file's path. The file has a .pf extension, so an example would be NOTEPAD.EXE-AF43252301.PF. An exception to the file name rule is the file that stores the boot's trace, which is always named NTOSBOOTB00DFAAD. PF (a convolution of the hexadecimal-compatible word BAADF00D, which programmers often use to represent uninitialized data).

Only after the Cache Manager has finished the boot trace (the time of which was defined earlier) does it collect page fault information for specific applications.

When the system boots or an application starts, the Cache Manager is called to give it an opportunity to perform Prefetching. The Cache Manager looks in the Prefetch directory to see if a trace file exists for the Prefetch scenario in question. If it does, the Cache Manager calls NTFS to Prefetch any MFT metadata file references, reads in the contents of each of the directories referenced, and finally opens each file referenced. It then calls the Memory Manager to read in any data and code specified in the trace that's not already in memory. The Memory Manager initiates all of the reads asynchronously and then waits for them to complete before letting an application's startup continue.

How does this scheme provide a performance benefit? The answer lies in the fact that during typical system boot or application startup, the order of faults is such that some pages are brought in from one part of a file, then from another part of the same file, then pages are read from a different file, then perhaps from a directory, and so on. This jumping around results in moving the heads around on the disk. Microsoft has learned through analysis that this slows boot and application startup times. By Prefetching data from a file or directory all at once before accessing another one, this scattered seeking for data on the disk is greatly reduced or eliminated, thus improving the overall time for system and application startup.

To minimize seeking even further, every three days or so, during system idle periods, the Task Scheduler organizes a list of files and directories in the order that they are referenced during a boot or application start, and stores the list in a file named \Windows\Prefech\Layout.ini.

LNK Shortcut Analyser

There are many references on the web to the file format of an LNK, and I have found the most informative to be the White Paper authored by Jesse Hager. The PDF can be downloaded at

http://www.i2s-lab.com/Papers/The_Windows_Shortcut_File_Format.pdf

The reason I carried out research into this area was due attempting to determine whether or not a user had accessed a CD which had CP images on it. I was able to see using WRA Streams MRU that the user had indeed accessed the CD, and in this case he used 4 different PCs to do so. What I needed to do was to see if there was a corroborative method to compliment WRA.

Rather than go through Jesse Hager's PDF in full, I am going to keep to a minimum the necessity to make reference to the PDF. By keeping this to a minimum, hopefully all users of the program should be able to replicate the programs output.

Due to actual Window dimensions I have not been able to incorporate the domain address, but I have included this in the breakdown.

The interesting fact about most LNK file is that they inherit the Creation Time and Last Accessed Date on each relevant folder in the full path. In the LNK on the following page, you can see the path:

C:\Documents and Settings\randy\Local Settings\Temporary Internet Files\Content.IE5
\O1QBCPMR\Rita's clouds.jpg

The LNK files have a makeup similar to C:/Windows (Creation Date and LA) Program Files (Creation Date and LA) Name of Program (Creation Date and LA).

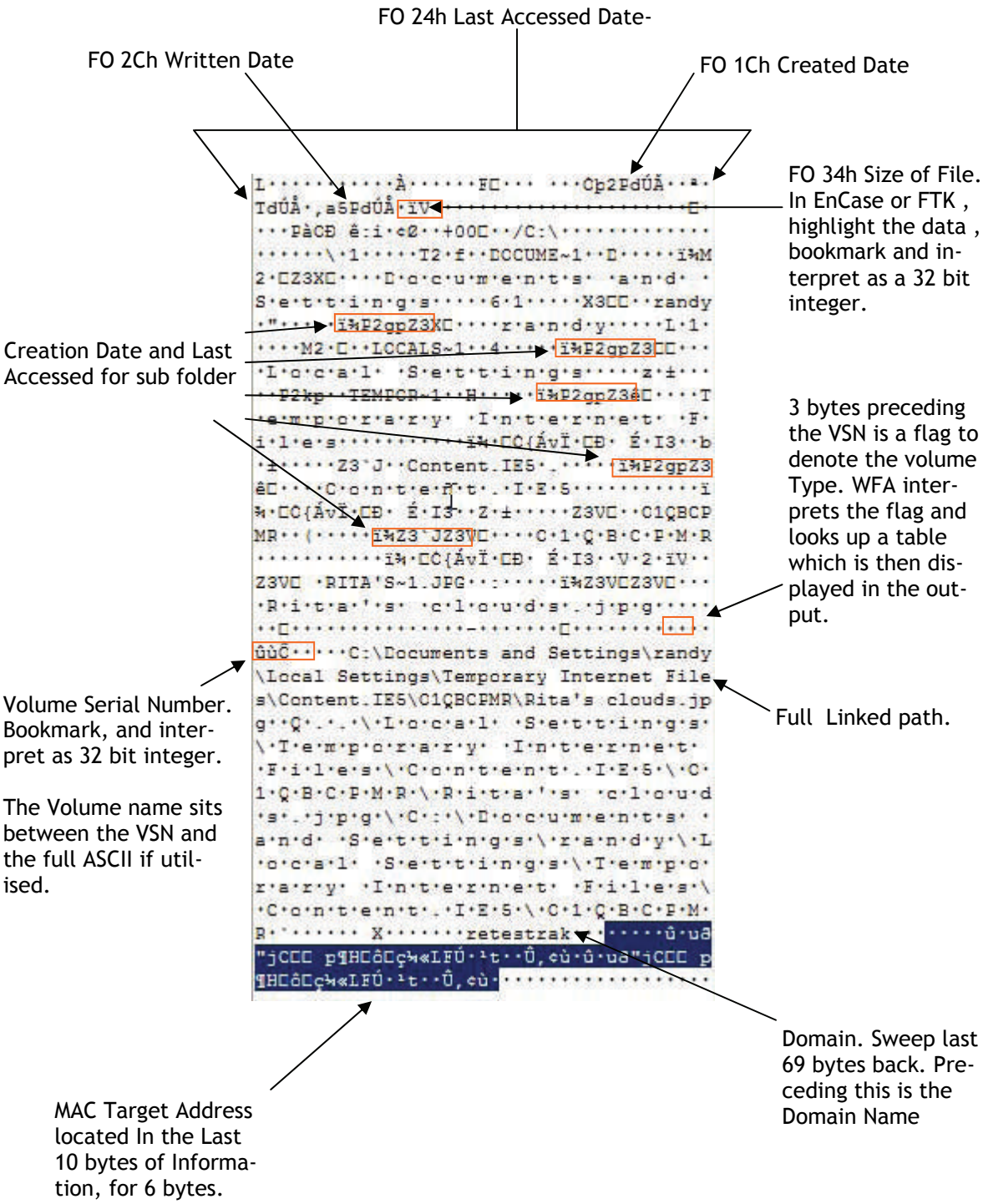
It is quite interesting when the LNK file is examined. References to files and folders within the LNK can on occasions be seen to no longer reside within the path of the LNK, which should arouse curiosity of file activity / movement.

The Last Accessed Time is not stamped in the subfolder structure within the LNK which is why in the output there are no times.

By going to the relevant offset in the LNK file, identifying the Timestamp requires the user to sweep over the first 4 bytes. This is the Creation Date and Last Access Time which defaults to 23:55. The next 4 bytes are the Last Access Date and Creation Time. Also, be aware that you may have to compensate for Time differences when interpreting the output.

LNK Shortcut Analyser

The dates commencing at FO 1Ch are the snapshot of the Creation Time , Last Accessed and Wriiten Dates for the file attributes in question, in this case Rita's Clouds.jpg



LNK Shortcut Analyser

Interesting Note

When I was doing some research work into LNK files I was trying to get hold of as many different OS LNK files to interpret.

I went onto a well known file sharing program and entered the search term 'LNK', and I was overwhelmed at the amount of files that were actually shared. To most it means nothing, however I became aware that some of the LNK files had 'notable' names particularly those with the second character being an '@'

I decided to download several of the LNK files including the 'notable' ones. I proceeded to process them through WFA and saw that the output told me the Computer Users Name, Volume Serial Number and Volume Name for the file in question where it was located at that time of the LNK creation.

Had I browsed the users shared files, I may have established whether they had indeed the 'notable' Files. That said, I think it would be corroborative evidence to suggest that you 'suspect' the path of the files involved, the Volume Serial Number, potential Volume Name , and depending on path (i.e. C/Documents and Settings/**user**/) the User Name, all before you go knocking on doors.

Allan S Hay

November 2005

ash368@btinternet.com