**De-mystifying Defrag: Identifying When Defrag Has Been Used for Anti-Forensics (Part 1 – Windows XP)**

By [Chad Tilbury](http://forensicmethods.com/author/chadtilbury) on August 5, 2009 in [Computer Forensics](http://forensicmethods.com/category/computer-forensics), [Incident Response](http://forensicmethods.com/category/incident-response), [Windows Registry](http://forensicmethods.com/category/windows-registry) — [Leave a comment](http://forensicmethods.com/defrag-xp#respond)

I have seen the following Windows Prefetch entries in nearly every Windows XP / Vista machine that I have reviewed over the past several years. Their existence always reminds me of the imperfect nature of information gained via individual artifacts. Does this mean that a user ran the Microsoft Defragmenter application on July 16, 2009 at 1:19PM? Or was the defragmenter started automatically by Windows? The defragmenter tool has been used very effectively as an anti-forensic tool since it was first introduced. In cases where data spoliation could be important, it is critical for the examiner to be able to identify any overt actions by a user.Complicating this is that starting with Windows XP, the operating system conducts limited defragmentation approximately every three days. [1] This post seeks to identify forensic artifacts which can help us determine if a user initiated the defrag application.

Figure 1: Defrag entries in C:\Windows\Prefetch directory

Figure 1: Defrag entries in C:\Windows\Prefetch directory

We will focus on two primary methods a user can invoke the Windows Defragmenter tool:

1. Running defragmenter from a graphical user interface (GUI)
2. Running defrag from the command line using defrag.exe

**Defragmenter Artifacts in Windows XP – Identifying GUI Usage**

The GUI defragmenter tool leaves a wealth of artifacts that can distinguish user execution of defrag from system execution. It is commonly accessed from the Start Menu -> Accessories -> System Tools menu. We will query the following artifacts to identify user actions:

1. Prefetch Entries
2. UserAssist Registry Key
3. Registry MMC Recent File list
4. File Access Timestamps

Prefetch Entries

The addition of Windows Prefetch in XP has provided investigators with an excellent artifact for identifying applications executed on a system. While it won’t give us everything we need in this situation, it is an excellent starting point. Entries are located in the C:\Windows\Prefetch directory and can be parsed using Mark McKinnon’s [Prefetch Parser](http://cfed-ttf.blogspot.com/2009/02/updated-prefetch-parser.html" \o "Prefetch Parser" \t "_blank) or your favorite forensic suite.

When the defragmenter is run using the GUI, only the dfrgntfs.exe entry is updated within the Prefetch directory (with an updated access time stamp and execution count). This immediately reveals that the artifacts shown in Figure 1 were not left by the GUI tool. It may also explain why we often see higher execution counts for dfrgntfs.exe than defrag.exe when parsing the Prefetch entries. As an aside, it is interesting to note that I found differences in how the execution count was updated.When using the GUI, the execution value for dfrgntfs.exe was incremented by one and when using the command line application, the counts were incremented by three.

Since the GUI version of the defragmenter is essentially a Microsoft Management Console (MMC) snap-in, an entry for MMC.exe is also created in the Prefetch folder.It is important to note that MMC.exe can be present in the Prefetch due to the use of other snap-ins (such as viewing the event logs). Its proximity to the dfrgntfs.exe entry is one clue, but Prefetch files can also show what was loaded by the application, and further investigation reveals that dfrgres.dll and dfrgui.dll are both loaded by MMC.exe whenever it facilitates the defragmenter snap-in.

Figure 2: Prefetch entries indicating the GUI defragmenter tool was run

Figure 2: Prefetch entries indicating the GUI defragmenter tool was run

UserAssist Registry Key

Prefetch can indicate that the GUI application was run, but it gives no information regarding user attribution. Luckily we have an artifact available in the NTUSER.dat hive file that does both. The UserAssist registry key, HKCU\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count, stores information on applications run *per user*. This is a terrific artifact for proving user activity and can be easily viewed using [UserAssist.exe](http://blog.didierstevens.com/2006/08/04/update-userassist-utility" \o "UserAssist Tool" \t "_blank)written by Didier Stevens. Evidence of manual execution can be found within this registry key when the defragmenter is accessed via the GUI interface. You should be looking for entries for “Disk Defragmenter.lnk” and “mmc.exe”. The “Last” time indicates when the application was last run by the user. One unfortunate limitation of this artifact is that applications run from the command line are not recorded.

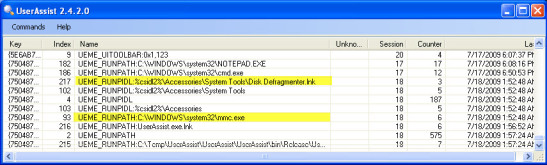
****

Figure 3: Using UserAssist.exe to parse the registry values

Registry MMC Recent File List

Since we know that the MMC is utilized by the defragmenter GUI, there is an additional registry artifact dedicated to recording MMC usage. Looking at HKCU\Software\Microsoft\Microsoft Management Console\Recent File List, we should see an entry for dfrg.msc, which is the Microsoft Common Console file for the defragmenter snap-in. If this is the last value recorded, the key last write time will indicate when the defragmenter was last run. Additionally, note that this registry key is under HKey\_Current\_User (NTUSER.dat hive), giving us another artifact for proving user attribution.

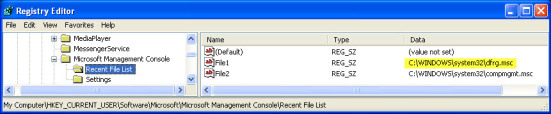


Figure 4: MMC Recent File List reviewed using regedit.exe

File Access Timestamps

Finally, we can tie everything together neatly by doing a timestamp analysis. The access timestamps for the following files indicate when these files were last run. Any mismatch with the times indicated in the Prefetch and UserAssist artifacts can tell us if there was any additional defrag activity after the GUI was run:

* C:\Windows\System32\Dfrgntfs.exe
* C:\Windows\System32\Dfrg.msc
* C:\Documents and Settings\All Users\Start Menu\Programs\Accessories\SystemTools\Disk Defragmenter.lnk

**Defragmenter Artifacts in Windows XP – Identifying Command Line Usage**

In contrast to our investigation of artifacts generated by the GUI interface, command line use of the defrag tool gives us much fewer artifacts to work with. We will be required to focus on:

1. Prefetch entries (including timestamps)
2. Timeline analysis of contemporaneous events
3. Layout.ini

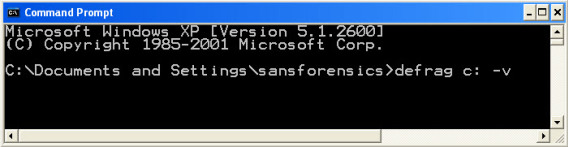
****

Figure 5: Example of running defrag.exe from the XP command line

Prefetch Entries

Unlike the artifacts for the GUI defragmenter, the Prefetch artifacts left by command line execution of defrag.exe are the same as those left by the Windows automated process. Upon command line execution both defrag.exe and dfrgntfs.exe are created in the Prefetch directory. Further, their last access times are updated to the time the application was run. This tells us when the defrag tool was last run, but does not allow us to differentiate between system defrags and user generated activity.Therefore we will need to turn to timeline analysis.

Timeline Analysis

With very limited artifacts, old fashioned timeline analysis will likely be our best bet to identify user defrag activity. This is not a theoretical exercise. We have seen instances of the defrag tool used as an anti-forensics tool in recent intrusion cases.Often this plays out with the intruder installing their payload on the system, deleting it, and then running defrag.exe to prevent the malware from being recovered by incident responders.



Figure 6: Sorting Prefetch entries by last accessed time and reviewing nearby applications

In this situation we will often be looking for the opening of a command shell (cmd.exe) near the time that defrag.exe was run. This is a key differentiator since Windows does not require an interactive command shell when kicking off automated processes. Figure 6 shows how this might look in the Prefetch directory. In addition to just Prefetch entries, other timeline entries like created and deleted files can provide further context. Admittedly, this evidence is very temporal and presumes that collection occurs as close a possible to the time the intrusion occurred.

Layout.ini

The layout.ini file is located in the C:\Windows\Prefetch directory and is used by the Prefetch process to more efficiently place system and frequently used applications during the limited defrag sequence. It is not used during a standard manual defragmentation. Therefore it can be a good indicator for distinguishing between user and system actions. In the example shown in Figure 7, the modification time of the layout.ini coincides with the defrag applications and indicates that they were run by the operating system (not the user).

Figure 7: Sequential prefetch entries indicating that a limited defrag was completed 

Figure 7: Sequential Prefetch entries indicating that a limited defrag was completed

A Note Regarding Event Logs

In Windows XP, there is no native capability to record defragmenter usage in the event logs. [2] Thus we will not be able to leverage this source of information as we could for other actions like task scheduler usage.

**Conclusion**

We stand our best chance of tying defragmenter execution to a user account if it was conducted via the GUI interface. Many of our most valuable Windows artifacts are not updated when an application is run from the command line. However, we will often be able to turn to old-fashioned timeline analysis to assist in these circumstances.

It should be noted that there are plenty of legitimate reasons for running the defragmenter tool. Other contemporaneous actions need to be reviewed to assess a user’s true intent.

**De-mystifying Defrag : Identifying When the Windows Defragmenter Has Been Used for Anti-Forensics (Part 2 – Vista / Windows 7)**

By [Chad Tilbury](http://forensicmethods.com/author/chadtilbury) on August 17, 2009 in [Computer Forensics](http://forensicmethods.com/category/computer-forensics), [Incident Response](http://forensicmethods.com/category/incident-response), [Windows Registry](http://forensicmethods.com/category/windows-registry) — [2 Comments](http://forensicmethods.com/defrag-win7#comments)

*Note: This post originally appeared on the SANS Forensics blog*

In[Part 1](http://forensicmethods.com/?p=138) of this post, we explored defragmenter usage in Windows XP, specifically trying to gain more information about user activity when we see the following in the Prefetch directory:

**Figure 1: Defrag entries shown from C:WindowsPrefetch directory**

Figure 1: Defrag entries shown from C:\\Windows\\Prefetch directory

Vista made many file system changes, modifying some of  the XP artifacts we relied upon in Part 1 and adding some artifacts that can greatly simplify our investigation.  Importantly, Vista ships with a default scheduled task for a full volume defragmentation every Wednesday evening at 1am.    This is in addition to the limited defrags conducted by the Prefetch / Superfetch components.   Thus we should expect to see even more defragmenter activity on a Vista machine.  Taking this into consideration, we will perform the same analysis that we did for Windows XP.

We will focus on the two primary methods a user can invoke the Windows Defragmenter tool:

1. Running defragmenter from a graphical user interface (GUI)
2. Running defrag from the command line using defrag.exe

**Defragmenter Artifacts in Vista – Identifying GUI Usage**

The Vista GUI defragmenter tool can be accessed in the same way it was in Windows XP (Start Menu -> Accessories -> System Tools ).  When performing a Vista analysis we will focus on the following artifacts:

1. Prefetch Entries
2. UserAssist Registry Key
3. File Timestamps
4. Event / Scheduler Logs

Prefetch Entries

The Prefetch signature generated by the GUI defragmenter is significantly different in Vista.  As shown in Figure 2, you should expect to see defrag.exe, dfrgntfs.exe and a new application named dfrgui.exe.  The latter is a replacement for the Microsoft Management Console (MMC) snap-in (dfrg.msc) that was used in XP.  Additionally, you may find consent.exe executed immediately before the GUI tool.  This is part of the Vista User Access Control (UAC) protections.

****

Figure 2: Prefetch Entries Present After GUI Defragmenter Launch

UserAssist Registry Key

Our XP analysis showed that we can use artifacts in the UserAssist key to show manual defragmenter execution by a user account.  For Vista, the artifact stored within UserAssist is for the dfrgui.lnk file, which is a shortcut to the dfrgui.exe application.  The “Last” time indicates when the application was last run by the user.

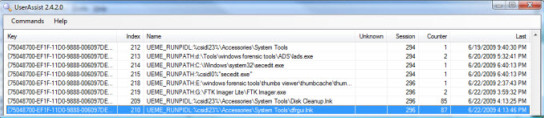
****

Figure 3: UserAssist Key Entry for Dfrgui.lnk

File  Timestamps

Vista ships by default with access timestamp updating disabled.  In systems where this has been enabled by the administrator, access timestamps can be used to identify when an executable was run and corroborate activity witnessed via other artifacts.

One important note regarding Prefetch entries is that in the absence of access timestamps, their modified timestamp can also indicate when the application was last run (due to the execution count being updated within the Prefetch file).

Event / Scheduler Logs

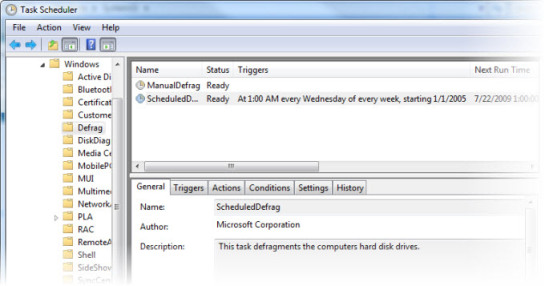
****

Figure 4: Task Scheduler for Defrag

Perhaps the most important change to affect our defragmenter analysis in Vista is that it now includes a mechanism to record defragmenter usage in the event logs.  Execution of the defragmenter is tracked for both manual and scheduled executions (see Figure 4).  This is part of the greatly expanded event logging accomplished by Vista.  Logging is on by default and can be reviewed either through the Task Scheduler “History” tab, or in the Application and Services Logs/Microsoft/Windows/Task Scheduler event log.  Figure 5 shows a log entry for a manual defrag task that was started by Vista.  The event logs provide enough granularity to identify when the defrag was scheduled, started, completed, and/or terminated before completion.  One limitation is that the defrag process is run as user SYSTEM, and thus user attribution is not possible using this artifact.  However,  it can be easily coupled with the UserAssist information covered previously to identify the user.  Unfortunately, command line and Prefetch/Superfetch (limited) defrag executions are not recorded in the event logs.

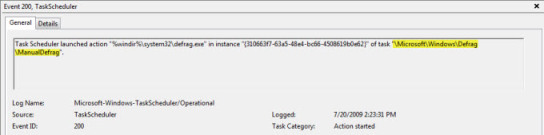
****

Figure 5: Event Log Entry Showing Manual Defrag

**Defragmenter Artifacts in Vista – Identifying Command Line Usage**

While we gained some additional artifacts in Vista for tracking GUI usage, we received no additional artifacts to track command line usage of the defragmenter tool.   Additionally, with access timestamps disabled by default, our timelines will be missing critical information making it more difficult to identify malicious use of the defragmenter.

As the steps are identical to those of XP, the reader is directed back to Part 1 for the discussion of artifacts generated by command line usage of the tool.

**Conclusion**

The expanded event logging within Vista provides an excellent means for examiners to identify manual versus scheduled executions of the defragmenter tool.   However, we will still need to use multiple artifacts to prove user attribution.  While identifying command line execution did not get any easier in Vista, it is still by no means a lost cause.  Good timeline analysis can often go a long way towards identifying user actions and intent.

**Computer Forensic Artifacts: Windows 7 Shellbags**

By [Chad Tilbury](http://forensicmethods.com/author/chadtilbury) on July 5, 2011 in [Computer Forensics](http://forensicmethods.com/category/computer-forensics),[Incident Response](http://forensicmethods.com/category/incident-response), [Tool Review](http://forensicmethods.com/category/tool-review), [Windows Registry](http://forensicmethods.com/category/windows-registry) — [Leave a comment](http://forensicmethods.com/shellbag#respond)

*Note: This post originally appeared on the SANS Forensics blog*

As Windows Registry artifacts go, the “Shellbag” keys tend to be some of the more complicated artifacts we have to decipher.  But they are worth the effort, giving an excellent means to prove the existence of files and folders along with user knowledge.  Shellbags can be used to answer the difficult questions of data enumeration in intrusion cases, identify the contents of long gone removable devices, and show the contents of previously mounted encrypted volumes.   Information persists for deleted folders, providing an invaluable reference for items no longer part of the file system.

**A Brief Overview**

Windows uses the Shellbag keys to store user preferences for GUI folder display within Windows Explorer.  Everything from visible columns to display mode (icons, details, list, etc.) to sort order are tracked.   If you have ever made changes to a folder and returned to that folder to find your new preferences intact, then you have seen Shellbags in action.  In the paper *Using shellbag information to reconstruct user activities*, the  authors write that “Shellbag information is available only for folders that have been opened and closed in Windows Explorer at least once” [1].  In other words, the simple existence of a Shellbag sub-key for a given directory indicates that the specific user account once visited that folder.  Thanks to the wonders of Windows Registry last write timestamps, we can also identify when that folder was first visited or last updated (and correlate with the embedded folder MAC times also stored by the key).  In some cases, historical file listings are available.  Given much of this information can only be found within Shellbag keys, it is little wonder why it has become a fan favorite.

**The Shift from Windows XP**

The architecture of Shellbag keys within Windows XP is well understood and has been broadly covered [1,2].  However this is not the case with the Windows 7 format.  I have recently had good luck using Shellbags within computer intrusion cases to show evidence of file system enumeration by attackers using compromised accounts.  These systems have largely been Windows XP or Server 2003 and when I first sat down to review a Windows 7 system I was severely disappointed.  Following the trend of many of our favorite Registry keys being updated in Windows 7, the Shellbag keys underwent a major transformation.  Gone are the familiar Shell, ShellNoRoam, and StreamMRU categories that respectively denoted network, local, and removable device folders.  Data from all of these locations still appears to be collected, but all three artifact categories are now stored within the Shell subkey.  The keys themselves are stored as a slightly different binary format making manual deciphering even more painful.  I was beating my head against the wall trying to reverse engineer the new format when Rob Lee suggested I do something really smart: leverage an existing tool and work backwards.  He introduced me to the[Tzworks Shellbag Parser](http://www.tzworks.net/prototype_page.php?proto_id=14) and I was hooked.  Besides being the only true Windows 7 Shellbags parser I am aware of, it does a remarkable job of parsing Shellbag structures.

**What you need to know**

Along with updating the Registry keys, Windows 7 also gave us a completely new user-specific Registry hive named **USRCLASS.dat**.  This hive supports the new User Access Control (UAC) and the mandatory access control integrity levels now baked into the operating system.  In oversimplified terms, it is used to record configuration information from user processes that do not have access to write to the standard registry hives.  In order to get all Shellbags information, we now need to parse both **NTUSER.da**t and **USRCLASS.dat** for each user account.  These Registry hives are located in the **%user profile%** and **%user profile%**\**AppData**\**Local**\**Microsoft\Windows** folders respectively.  The specific Shellbag keys  are:

USRCLASS.DAT\Local Settings\Software\Microsoft\Windows\Shell\BagMRU

USRCLASS.DAT\Local Settings\Software\Microsoft\Windows\Shell\Bags

NTUSER.DAT\Software\Microsoft\Windows\Shell\BagMRU

NTUSER.DAT\Software\Microsoft\Windows\Shell\Bags

Microsoft documents additional Shellbag keys that may be present on Win7 systems, but after a review of several Win7, Vista, and 2008R2 systems I have been unable to find any evidence of them being used [4].  For reference purposes, the additional keys are (Wow6432Node keys are located on x64 systems):

NTUSER.DAT \Software\Microsoft\Windows\ShellNoRoam\BagMRU

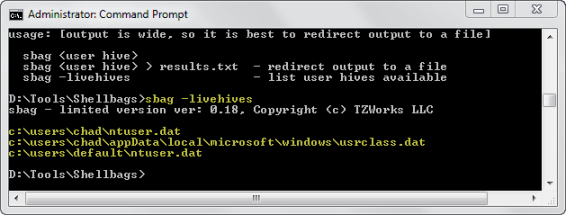
NTUSER.DAT\Software\Microsoft\Windows\ShellNoRoam\Bags

USRCLASS.DAT\Wow6432Node\Local Settings\Software\Microsoft\Windows\Shell\Bags

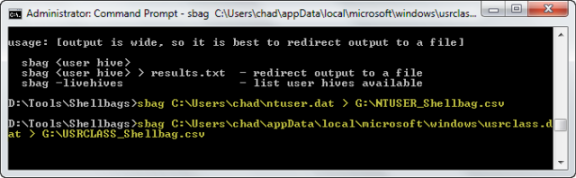
USRCLASS.DAT\Wow6432Node\Local Settings\Software

**Parsing Shellbag Keys Using TZworks Sbag.exe**

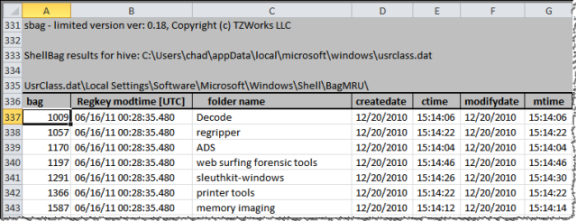
Using the TZworks tool to parse Shellbags is trivial.  It is a command-line tool with just a few parameters.  If you are working on a live system, you can use the “listhives” parameter to have it identify the available user Registry hives.

[](http://i1.wp.com/blogs.sans.org/computer-forensics/files/2011/06/sbag_livehives2.png)

Once you know the paths of the hives you wish to parse,  execute Sbag.exe and redirect the results to a .csv file (results are “|” delimited).

[](http://i2.wp.com/blogs.sans.org/computer-forensics/files/2011/06/sbag2.png)

Finally, import the files into your favorite spreadsheet and start your analysis.

[](http://i1.wp.com/blogs.sans.org/computer-forensics/files/2011/06/sbag_spreadsheet3.png)

What information should you expect to gather from the Shellbags keys?  Each folder will have:

* **Bag Number** [Identifies the “Bags” subkey containing user preferences – aka Nodeslot]
* **Registry key last write time**[First access of folder or last preference change]
* **Folder name**
* **Full path**
* **Embedded creation date / time** [Stored at the time the BagMRU key was created]
* **Embedded modify date / time** [Stored at the time the BagMRU key was created]
* **Embedded access date / time** [Stored at the time the BagMRU key was created]

My initial testing of TZworks Sbag.exe has found it to be highly reliable.  In comparison to my previous go-to tool, Windows Registry Analyzer (which only accurately parses XP Shellbags), it does a more complete job, particularly with regard to timestamps.  It works with both  XP and Windows 7 artifacts, can parse both live and exported Registry hives, and the output is extremely easy to work with.  Versions for Windows, Linux, and Mac OS X are available.  If you haven’t incorporated Shellbag review into your examinations, now is the time!  Also, keep in mind that XP and Windows 7 Shellbag analysis is now included  in the SANS 508 Advanced Forensics class.