# The Windows Forensic Journey

**Version 2.0**
**April-2025**


## By Dr. Shlomi Boutnaru


Created using Craiyon, AI Image Generator

# Table of Contents

# Introduction

When using a workstation/server running a Microsoft Windows based operating system there are different forensics artifacts which are created. I have decided to write a series of short writeups aimed at providing the basic understanding on the different forensics artifacts created by Windows.

Overall, I wanted to create something that will improve the overall knowledge of digital forensics in Windows with writeups that can be read in 1-3 mins. I hope you are going to enjoy the ride.

Lastly, you can follow me on twitter - @boutnaru (https://twitter.com/boutnaru). Also, you can read my other writeups on medium - https://medium.com/@boutnaru. Lastly, You can find my free eBooks at https://TheLearningJourneyEbooks.com.

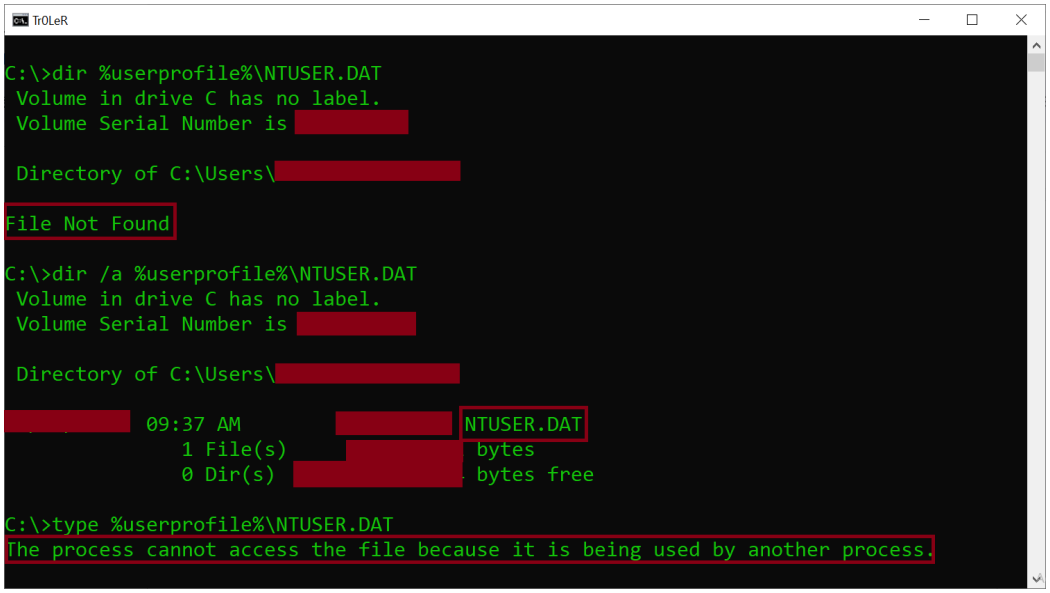
Lets GO!!!!!!

# NTUSER.DAT

The NTUSER.DAT contains user account settings and customizations of a specific Windows user (which can be a local user or a domain user), think about the wallpaper settings as an example or the preferred keyboard layout. It is created by the operating system the first time a user logs on the system. The file is located in the user profile directory of the user "%userprofile%\NTUSER.DAT"[1].

Overall, the file is hidden thus we can see it using the "/a" flag of "dir" which is a builtin command of cmd.exe[2] - as shown in the screenshot below. The "NTUSER.DAT" is basically a registry hive[3] which is loaded to "HKEY_USERS" and is pointed to by "HKEY_CURRENT_USER" when the user logs on to the system. We can use the "NTUSER.DAT" file for offline analysis on a non-running system.

Lastly, there are also backups and transaction logs for the "NTUSER.DATA" (also stored in the %userprofile% directory with extensions like ".log"). The "ntuser.ini" file describes roaming profiles used in networked environment[4]. As with the files of the system's registry ("%windir%\system32\config"), both "NTUSER.DAT" and its related files are opened exclusively by the operating system when the user is logged on.



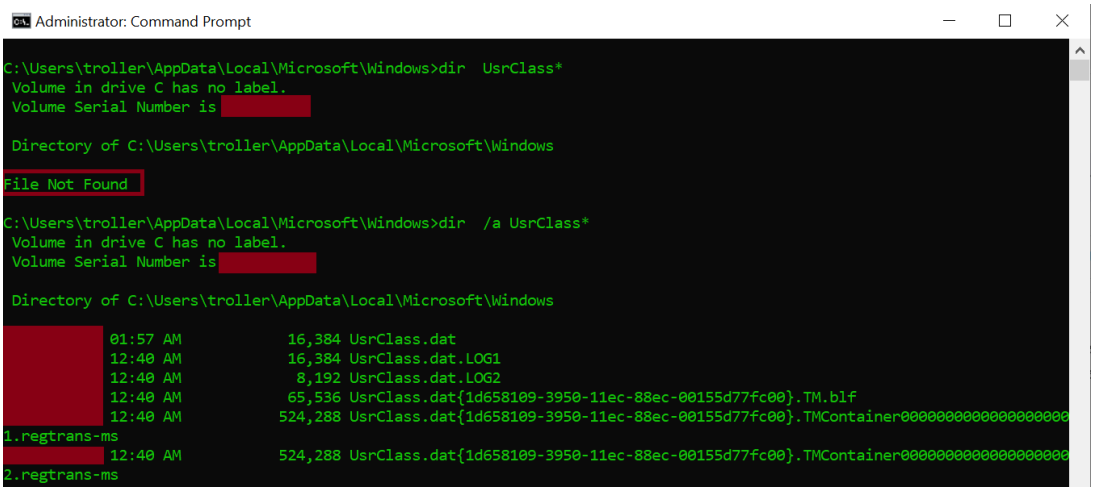

[1] https://appuals.com/ntuser-dat-file-explained/
[2] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b
[3] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[4] https://www.techtarget.com/searchenterprisedesktop/blog/Windows-Enterprise-Desktop/Understanding-NTUserdat-in-Windows-10

# UsrClass.dat

The “UsrClass.dat” file is located “C:\Users\%username%\AppData\Local\Microsoft\Windows” (which can be accessed also by “%userprofile%\AppData\Local\Microsoft\Windows”). “UsrClass.dat” is a registry hive file. Together with the “NTUSER.DAT”[5] registry hive file it composes the registry hive of a logged on user (created under HKEY_USERS and pointed by HKEY_CURRENT_USER).

Moreover, like with “NTUSER.DAT” there are backups\transactions files. By the way, the “UsrClass.dat” is also a hidden file - as shown in the screenshot below. In case we create a new local user account and perform a secondary logon using “runas.exe”[6] a “UsrClass.dat” file is created. The newly created registry file contains default subkeys like “CLSID” and “Local Settings”.

Lastly, “UsrClass.dat” contains highly valuable forensics artifacts like “ShellBags”. Those are records of the user’s view settings and preferences while exploring folders[7].




[5] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[6] https://medium.com/@boutnaru/the-windows-process-journey-runas-exe-run-as-utility-3c1e0b8aaa67
[7] https://forensafe.com/blogs/shellbags.html

# LNK Files (Shortcut Files)

Overall, users/the OS can create shortcuts to files/directories. We can think of a shortcut as a file which contains information used for accessing another file/folder. By default, Windows' shortcut files have a "*.lnk" extension (cause they are link files). Windows creates LNK files automatically when users open non-executable files, we can think about documents and images for example[8].

Moreover, LNK files contain different types of attributes (not all of that is displayed in the GUI of Windows) - as shown in the screenshot below. Among the information we can find: the size of the target file, timestamps (both for the LNK file and the target file), the system name, volume serial number, MAC address, indication if the target file is stored local/remote and attributes of the target file (readonly/hidden/etc). There is a great tool by Eric Zimmerman called LECmd[9] which parses LNK files - as shown in the screenshot below in an XML output (it shows more information than the GUI). Lastly, LNK files is based on the "Shell Link Binary File Format"[10].

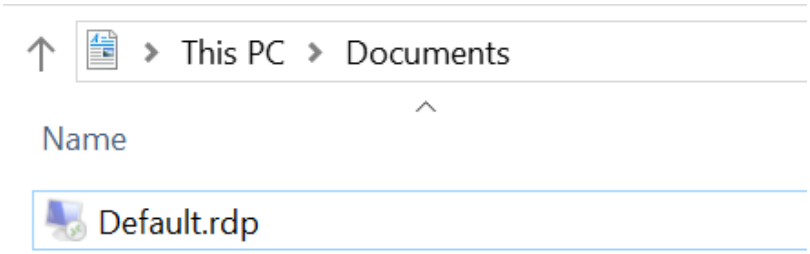

[8] https://dfir.pubpub.org/pub/wfuxlu9v/release/1
[9] https://ericzimmerman.github.io/#!index.md
[10] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-shllink/16cb4ca1-9339-4d0c-a68d-bf1d6cc0f943

# RDP Connection Settings

When using the "mstsc.exe" utility[11] for connecting remotely to Windows systems (workstation/server) the client saves the connection settings to a hidden "Default.rdp" file[12].

Overall, the "Default.rdp" file is stored for each user as a part of the "Documents" folder[13] - as shown in the screenshot below. By the way, we can change the name of the file after creation if we want and even create our own connection settings files (which is basically a text file with configurations).

Lastly, among the settings that can be stored in an "*.rdp" file we can find: server information, user information (including an encrypted password), display settings (like screen size and color depth), enabling connection bar, customizing local resources access (such as remote audio, keyboard input behavior, printer access, clipboard sharing, share local drives), optimizing performance (for example connection speed and persistent bitmap caching) and more[14].




[11] https://medium.com/@boutnaru/the-windows-process-journey-mstsc-exe-remote-desktop-connection-981bae774bac
[12] https://blog.devolutions.net/2025/03/using-rdp-without-leaving-traces-the-mstsc-public-mode/
[13] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/mstsc
[14] https://v2cloud.com/blog/how-to-create-open-and-configure-rdp-files

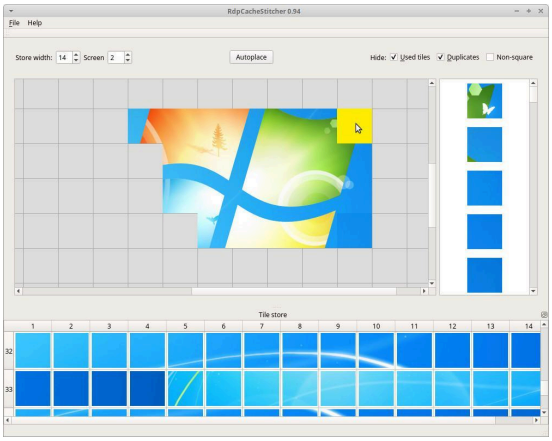
# RDP Bitmap Cache (Remote Desktop Protocol Bitmap Cache)

When using “mstsc.exe”[15] for connecting remotely to Windows systems (workstation/server) the client leverages an RDP caching mechanism. It is used to reduce the amount of data to be sent by the server. The caching is done by caching those parts of the screen that have not changed since the display was last refreshed[16].

Thus, when enabled the RDP bitmap caching allows the session to use data already in the local cache files to provide better experience and reduce network bandwidth. Each bitmap cache entry stores bitmap data and metadata (color depth, key and dimensions). It is important to understand that this cache is persistent even after the RDP session has been closed[17].

Moreover, the cache files store raw bitmaps in the forms of tiles. Although the tile size can vary, the most common size is 64x64 pixels. The location of the RDP bitmap cache is “%localappdata%\Microsoft\Terminal Server Client\Cache” (as a reminder “Terminal Server” is the old RDP name). There we can have two type of files “bcacheX.bmc” (where X is 2/22/24 which represent the quality) and “CacheXYZW.bin” (where XYZW are numbers that are generated on each session), we can use their timestamp to correlate with other log files[18].

Lastly, we can use the open source “BMC-Tools” (which is written in Python) in order to parse the RDP bitmap cache[19]. Also, we can use the perl script in order to try and rebuild some of the screenshots automatically[20] after they are extracted by using “BMC-Tools”. There is also an option of trying to stitch the bitmaps using a UI tool called “RdpCacheStitcher”[21] - shown below.



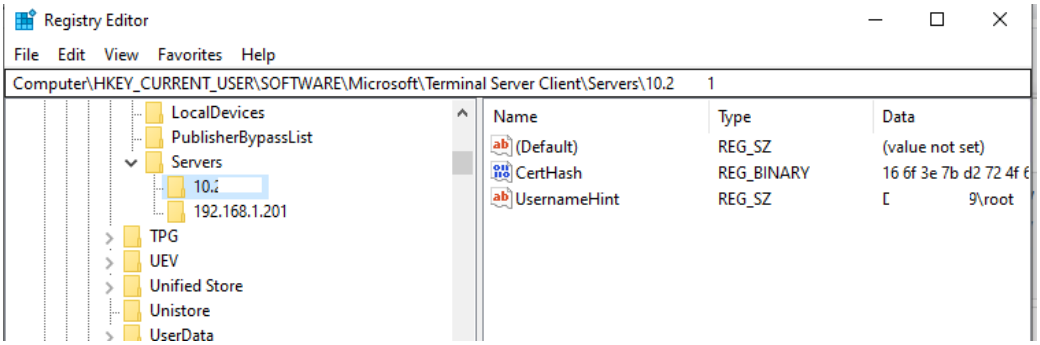

[15] https://medium.com/@boutnaru/the-windows-process-journey-mstsc-exe-remote-desktop-connection-981bae774bac
[16] https://security.opentext.com/appDetails/RDP-Cached-Bitmap-Extractor
[17] https://www.paloaltonetworks.com/blog/security-operations/playbook-of-the-week-uncover-your-rdp-secrets/
[18] https://www.linkedin.com/pulse/blind-forensics-rdp-bitmap-cache-ronald-craft
[19] https://github.com/ANSSI-FR/bmc-tools
[20] https://github.com/brimorlabs/rdpieces
[21] https://github.com/BSI-Bund/RdpCacheStitcher

# Username Hint

One of the side effects of using the "mstsc.exe" utility[22] for connecting remotely to a server/workstation is the "UsernameHint". It is used for speeding up the login flow. This is done by remembering the last username used to connect to a specific server [23]

Overall, the information is stored as part of the registry[24] in the following location: "HKCU\Software\Microsoft\Terminal Server Client\Servers\[SERVER]\UsernameHint", where "[SERVER]" is a place order for the IP\hostname of the remote RDP server[25] - as shown in the screenshot below[26].

Lastly, for a reference implementation of using "UsernameHint" we can checkout the source code (the "LoadUsernameHint" function) as part of ReactOS [27].



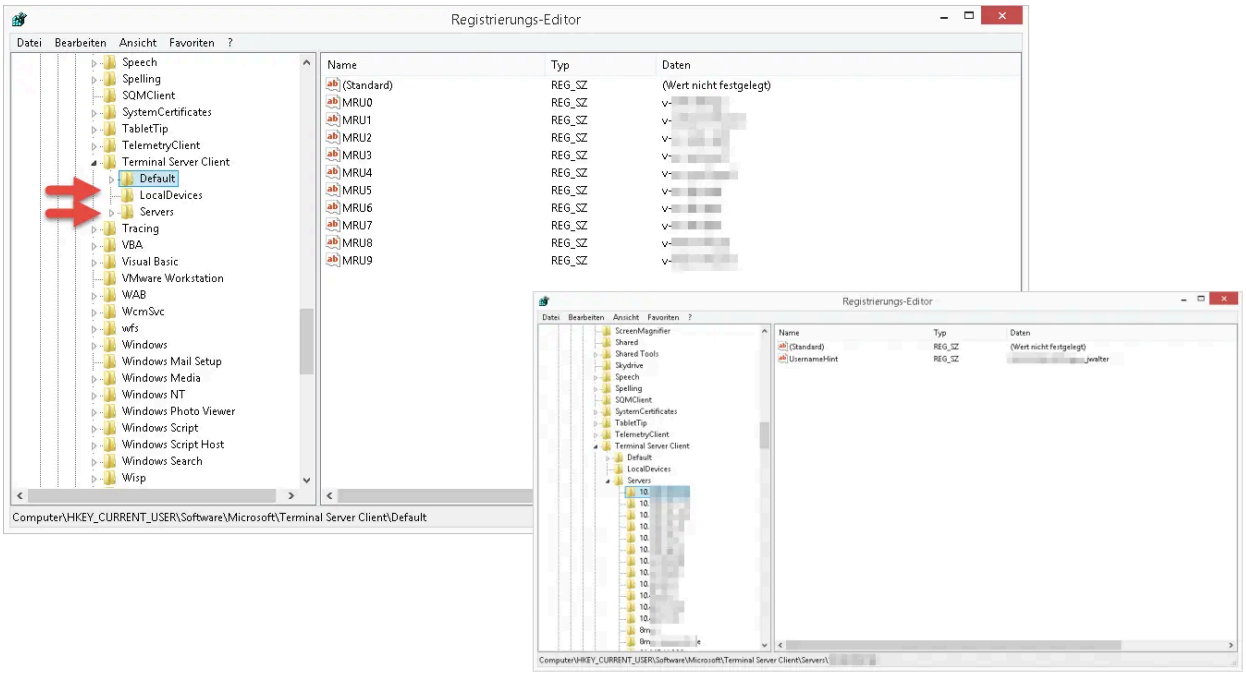

[22] https://medium.com/@boutnaru/the-windows-process-journey-mstsc-exe-remote-desktop-connection-981bae774bac
[23] https://blog.devolutions.net/2025/03/using-rdp-without-leaving-traces-the-mstsc-public-mode/
[24] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[25] https://learn.microsoft.com/en-us/answers/questions/208750/retrieve-ip-address-that-we-already-rdp-in-our-mac
[26] https://woshub.com/securing-rdp-connections-trusted-ssl-tls-certificates/
[27] https://github.com/reactos/reactos/blob/master/base/applications/mstsc/connectdialog.c

# RDP Connection History (Remote Desktop Protocol Connection History)

When using “mstsc.exe”[28] for initiating an RDP connection, every successful connection causes the connection details to be logged (IP/hostname information). This information is saved for each user in the following registry branch: “HKCU\SOFTWARE\Microsoft\Terminal Server Client”. There are two relevant registry keys: “Default” and “Servers”[29].

Moreover, “Default” holds the history of the last 10 RDP connections. While “Servers” contains a list of all RDP connections that have ever been created from the local machine by the user. An example of both is shown in the screenshots below. By the way, MRU shown in the screenshots stands for “Most Recently Used”[30].

Lastly, when using “mstsc.exe” a hidden file named “Default.rdp” is created in the home directory of the user, the full path is “%homepath%\Documents\Default.rdp”[31].



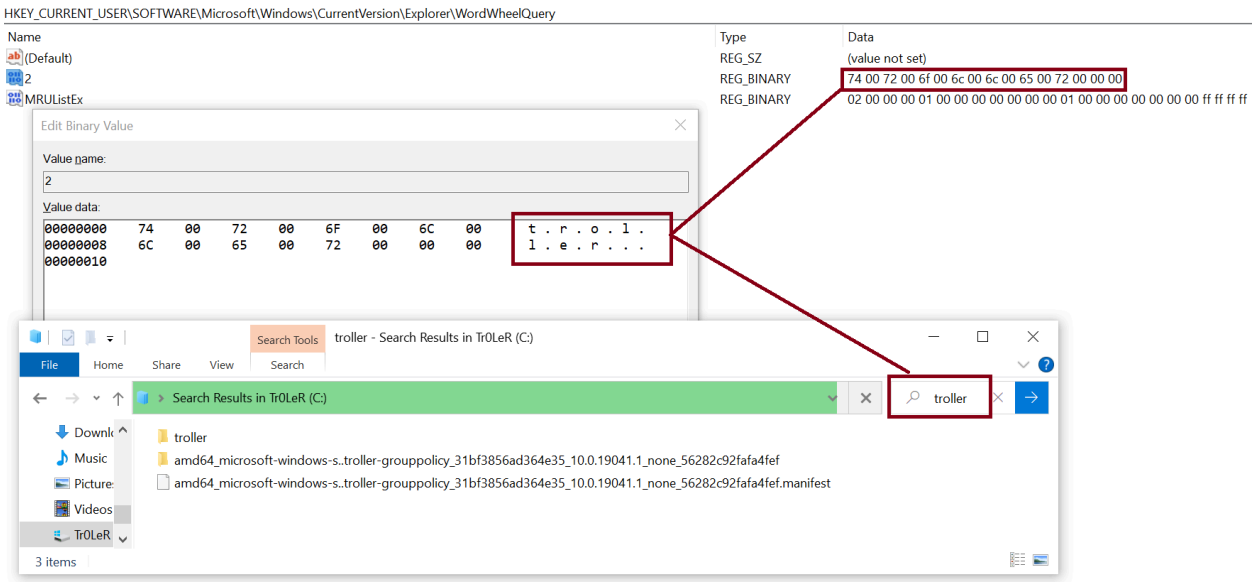

[28] https://medium.com/@boutnaru/the-windows-process-journey-mstsc-exe-remote-desktop-connection-981bae774bac
[29] https://www.tachytelic.net/2019/01/clear-rdp-cache/
[30] https://www.fity.club/lists/suggestions/hkey-current-user-software-microsoft-windows/
[31] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/mstsc

# Word Wheel Query (File Explorer Searches)

In case users are using the builtin search feature in "File Explorer"[32] we can extract the searched items for the "WordWheelQuery" registry key - as shown in the screenshot below. This is relevant for different versions of Windows such as 7/8/10/11[33].

Overall, we can read the "WordWheelQuery" registry key from the following location: "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\WordWheelQuery" in case of an online analysis (when the operating system is running). In case of an offline analysis we can extract the information from the NTUSER.DAT file (which holds the information/configurations of a specific local/domain Windows user).

Lastly, by removing the data values in the location mentioned above we basically erase/clear the search history in "File Explorer"[34].




[32] https://medium.com/@boutnaru/the-windows-concept-journey-file-explorer-previously-windows-explorer-e48077b135a0
[33] https://forensafe.com/blogs/searchedstrings.html
[34] https://www.windowscentral.com/how-clear-search-history-file-explorer-windows-10

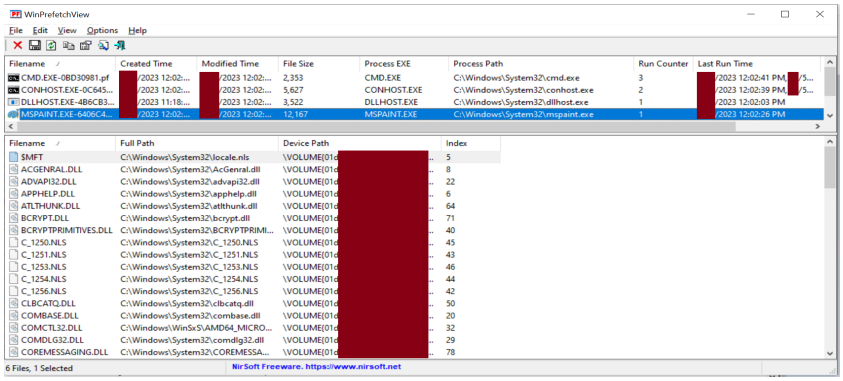
# Prefetch

Since Windows XP there is a component called “Prefetcher” which is part of the Memory Manager. Its goal is to speed up the Windows boot process and reduce the time it takes to start programs. This is done by caching to RAM files that are needed while the program is launched (based on information collected from previous executions). Since Windows Vista this mechanism was extended by “SuperFetch” and “ReadyBoost”[35].

Overall, for every process execution there is a creation/modification of a “*.pf” file in the “%systemroot%\Prefetch” directory. It is important to know that those files are not user-specific and have a global scope. Due to that, there is no user information as part of the artifact. The existence of a “*.pf” file states that a certain executable was launched on the system[36].

Moreover, from prefetch files we can extract the following information: file size, the binary name, the number of times the binary was executed, the path to the binary, first execution time, last execution time (up to the last 8) and a list of referenced files (like “*.dll” files that have been loaded by the process). We can use “WinPrefetchView” by Nirsoft[37] for parsing the information of “*.pf” file - as shown in the screenshot below.

Lastly, the patten of the “*.pf” files’ names is “[ORIGINAL_BINARY_NAME]-[HASH_OF_APP_PATH].pf”, an example of that is “MSPAINT.EXE-6406C4A1.pf”[38]. For disabling prefetch we need to set the value name “EnablePrefetcher” to the value “0” in the following registry key “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters”[39] By the way, the prefetch technology is based on a patent from Microsoft[40].



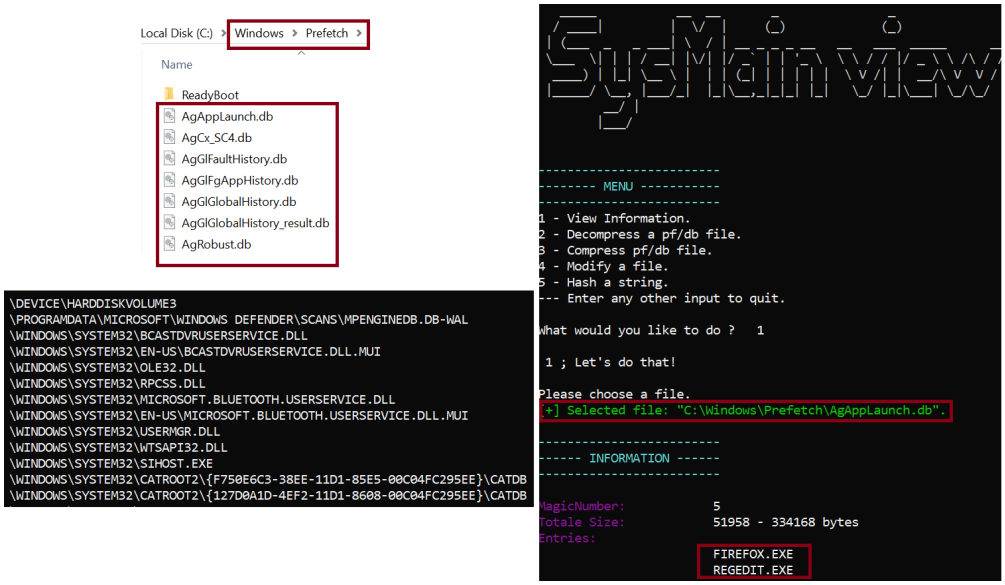

[35] https://en.wikipedia.org/wiki/Prefetcher
[36] https://www.hackthebox.com/blog/how-to-detect-psexec-and-lateral-movements
[37] https://www.nirsoft.net/utils/win_prefetch_view.html
[38] https://docs.velociraptor.app/docs/forensic/evidence_of_execution/
[39] https://4n6shetty.com/How-Windows-Artifact-Prefetch-Can-Help-in-Digital-Forensics-Investigations-in-Windows-11-Machine
[40] https://patents.google.com/patent/US6317818B1/en

# SuperFetch

“SuperFetch” is an extension of the “Prefetch” feature[41], which is part of the Windows operating system. Its goal is to proactively optimize application memory with regards to time and usage scenarios (it could be that applications used in the morning are different from those after launch). “SuperFetch” tracks "performance scenarios" and is specifically designed to anticipate frequently run applications after system activity like standby mode, hibernation, and fast-user switching. Thus, allowing it to model user behavior and make better decisions about when to pre-load application data into memory[42].

Moreover, as opposed to “Prefetch” which uses “*.pf” files “SuperFetch” uses “Ag*.db” files. Those files are also stored at “%windir%\Prefetch” - as shown in the screenshot below. Among the data stored by “SuperFetch” we can find: application executable name, execution count, foreground count, timeframes of executions, a list of files mapped by applications (like DLLs, db files, documents and more). Since Windows 10 the service[43] which is responsible for “SuperFetch” is “SysMain”[44].

Lastly, the “SysMain” service is implemented in the “%windir%\system32\sysmain.dll” file, which is hosted by the “svchost.exe”[45] when the service is executed. We can use the “SysMainView” for parsing both “*.pf” files and SuperFetch “*.db” files[46] - as shown in the screenshot below. By the way, there are also UI based tools like “Prefetch-Browser”[47].



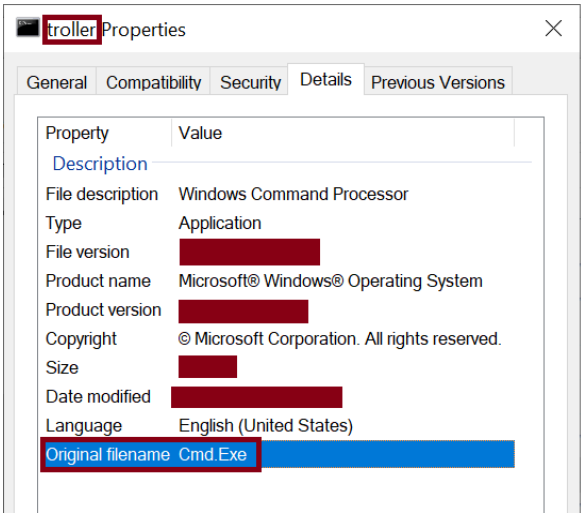

[41] https://medium.com/@boutnaru/the-windows-forensics-journey-prefetch-59af4722ceb9
[42] https://www.sans.org/blog/what-is-new-in-windows-application-execution/
[43] https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4
[44] https://www.technipages.com/windows-enable-disable-superfetch/
[45] https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f
[46] https://github.com/MathildeVenault/SysMainView
[47] https://github.com/kacos2000/Prefetch-Browser

# OriginalFileName (Original File Name Field)

"OriginalFileName" is a field as part of the version information[48], which is part of the PE (Portable Executable) header[49]. It can be used for getting the name the file was created with[50].

Overall, in case we have a signed file like: "svchost.exe"[51] or "cmd.exe"[52] changing the "OriginalFileName" can leave a trail[53]. Thus, modifying the "OriginalFileName" causes the digital signature to be invalid.

Lastly, different security products/features can configure rules based on the "original filename" (and not just based on the current filename) such as "Applocker"[54]. Because the "OriginalFileName" field is part of the "Version Information" data it is stored as a resource of the executable[55].



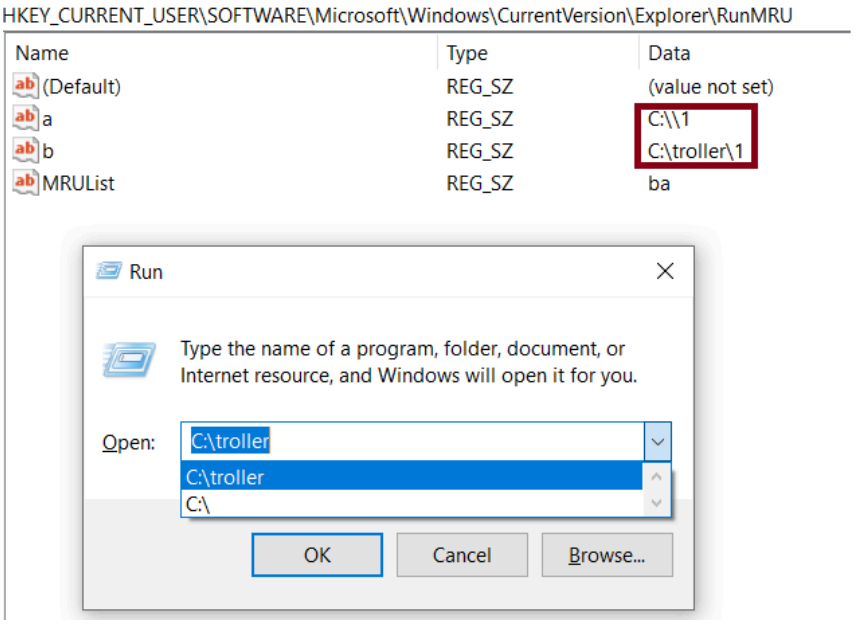

[48] https://www.herongyang.com/C-Sharp/FileVersionInfo-What-Is-FileVersionInfo.html
[49] https://medium.com/@boutnaru/the-portable-executable-journey-dos-header-ea5b29f15612
[50] https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics.fileversioninfo.originalfilename?view=net-9.0
[51] https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f
[52] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b
[53] https://stackoverflow.com/questions/72895583/is-there-an-elf-equivelant-of-pe-original-filename
[54] https://learn.microsoft.com/en-us/windows/security/application-security/application-control/app-control-for-business/applocker/working-with-applocker-rules
[55] https://medium.com/@boutnaru/the-windows-portable-executable-journey-version-information-01c29fe7cdb2

# Run MRU (Run Dialog Box Most Recently Used)

When using the "Run" command box ("Winkey+R") users can directly launch programs or open files/folders. "Run" includes a dropdown list of the last commands executed - as shown in the screenshot below. Those commands are saved in the registry under the "RunMRU" key[56] MRU in that case stands for "Most Recently Used.

Overall, "RunMRU" is saved separately for each Windows user (local/domain) in the following registry location: "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU" which we can access while the operating system is running (online analysis). For an offline analysis we can read the information for the NTUSER.DAT file ("Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU").

Moreover, each command is saved in a different value and the "MRUList" contains a list of all the commands to show and in what order. Also, each command is saved with a suffix with "\1" - as shown in the screenshot below. We can also clear the "RunMRU" history by removing the keys and values detailed above[57]. Lastly, "RunMRU" is not the MRU list in Windows there are others like "Microsoft Office MRU".



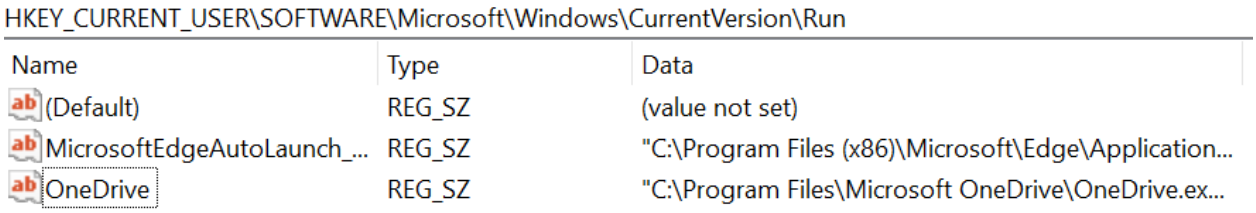

[56] https://forensafe.com/blogs/runmrukey.html
[57] https://www.thewindowsclub.com/clear-most-recently-used-mru-list?expand_article=1

# Run (Registry Key)

The registry has two relevant locations for the "Run" key[58]. Those are: "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" and "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run". The run registry key is used for configuring a list of programs\applications to execute when the user logs on.

Overall, each data value (as part of the run key) is the command line to execute (can also include arguments) and it is limited to 260 characters - as shown in the screenshot below. In case multiple programs\applications are registered the order of execution is indeterminate[59].

Lastly, entries as part of the "HKEY_LOCAL_MACHINE" run key will execute every time any user logs in to the system. On the other hand, entries part of the "HKEY_CURRENT_USER" run key are executed every time a specific user logs to the system[60].



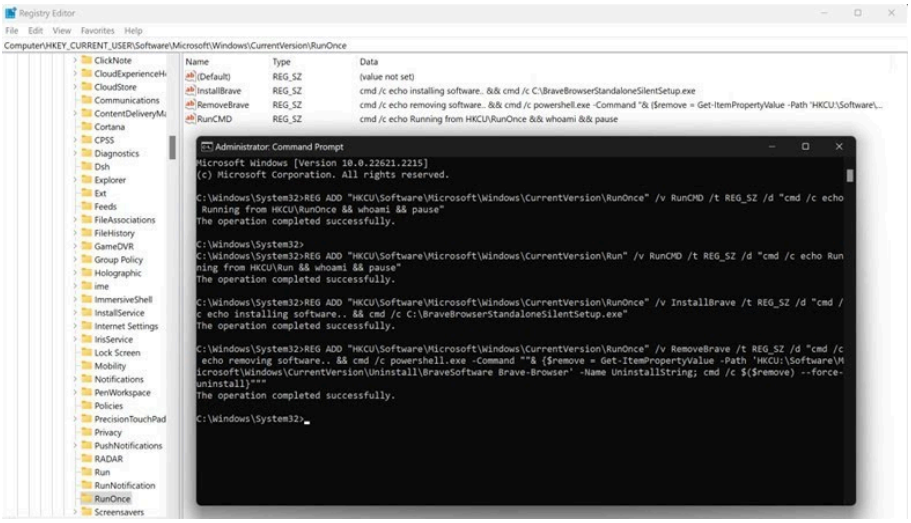

[58] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[59] https://learn.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys
[60] https://www.alkanesolutions.co.uk/2023/08/31/run-an-executable-after-windows-logon/

# RunOnce (Registry Key)

The registry has two relevant locations for the "RunOnce" key[61]. Those are: "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" and "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce". The run registry key is used for configuring a list of programs\applications to execute when the user logs on - as shown in the screenshot below[62].

Overall, each data value (as part of the run key) is the command line to execute (can also include arguments) and it is limited to 260 characters - as shown in the screenshot below. In case multiple programs\applications are registered the order of execution is indeterminate[63].

Lastly, entries as part of the "HKEY_LOCAL_MACHINE" run key will execute once (the registry value will be deleted prior to the command line being run) for the first user (any user) that logs in to the system. On the other hand, entries part of the "HKEY_CURRENT_USER" run key are executed once when a specific user logs to the system. Also, because the command can fail to run (and won't run again) we can prefix the registry's value name with "!" which ensures the command is run successfully[64].




[61] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[62] https://www.youtube.com/watch?v=zgFzCq5uEPw
[63] https://learn.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys
[64] https://www.alkanesolutions.co.uk/2023/08/31/run-an-executable-after-windows-logon/

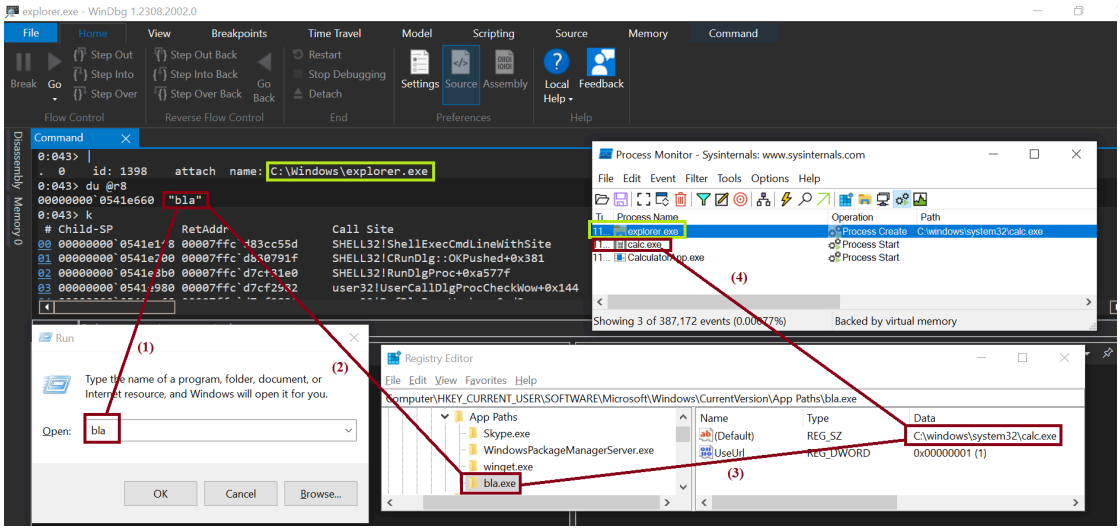
# App Paths (Application Registration)

Application Registration (aka “App Paths”) is a registry[65] key used Windows in order to provide a private search path for specific “*.exe”/”*.dll” files. The location of the key is the following: “HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\”[66]. We also have a counterpart in “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\”[67].

Thus, we can say that “App Paths” has two major goals. First, to map an executable name (like “App.exe”) to the program's fully qualified path. Second, appending information to the PATH environment variable. It is important to understand that executable registered as a subkey can be launched using the “start” command in “cmd.exe” even if they are not found using PATH [68].

Overall, “App Paths” is checked as part of the flow of “ShellExecute”\”ShellExecuteEx”[69] that is for finding the current application - as shown in the screenshot below. We can review that in the reference implementation as part of ReacOS[70].

Lastly, when creating a sub key as part of the application registration we can use one of 6 entries: “(Default)” (the full path of the application), “DontUseDesktopChangeRouter” (which have to be used in a case of a debugger to avoid deadlocks), “DropTarget” (the CLSID of an object that implements IDropTarget), “Path” (string to append to the PATH environment variable), “SupportedProtocols” (URL protocol schemes for a given key) and “UseUrl” (indicates that your application can accept a URL)[71].




[65] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[66] https://docs.revenera.com/installshield28helplib/helplibrary/PA_AppPaths.htm
[67] https://helgeklein.com/blog/how-the-app-paths-registry-key-makes-windows-both-faster-and-safer/
[68] https://renenyffenegger.ch/notes/Windows/registry/tree/HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/CurrentVersion/App-Paths/index
[69] https://learn.microsoft.com/en-us/windows/win32/shell/launch
[70] https://github.com/reactos/reactos/blob/master/dll/win32/shell32/shlexec.cpp#L762
[71] https://learn.microsoft.com/en-us/windows/win32/shell/app-registration

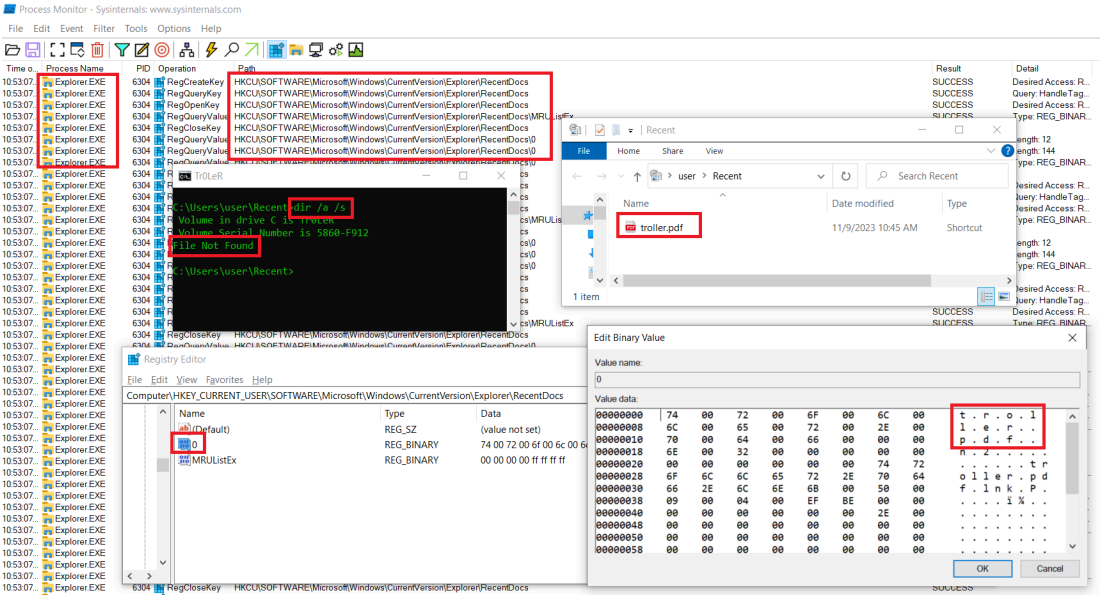
# Recent Docs (Recently Used Documents)

Overall, "RecentDocs" is a key in the registry located at the following location: "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs". We can see there a list of recently accessed files (documents/images/presentations/links/etc).

Thus, this key holds a list of accessed files using "File Explorer"[72] by the currently logged on user. Those are basically the entries we see if we open the directory "C:\users\%username%\Recent" using "File Explorer"[73]. By the way, we can also use "%userprofile%\recent".

However, if we open the "C:\users%username%\Recent" folder using "cmd.exe"[74] and list for files (including hidden files) we won't see any file there - as shown in the screenshot below. By the way, the RecentDocs key also has an MRU list (MRUListEx) of type REG_BINARY, which gives the access order of the files[75].

Moreover, opening the same directory with "File Explorer" will show up the entries. Using "Process Monitor" we can verify that "explorer.exe"[76] parses the "RecentDocs" registry key in order to show files accessed - as shown in the screenshot below. Lastly, in "RecentDocs" has some correlation with the "%appdata%\Microsoft\Windows\Recent" - more on that in a separate writeup.



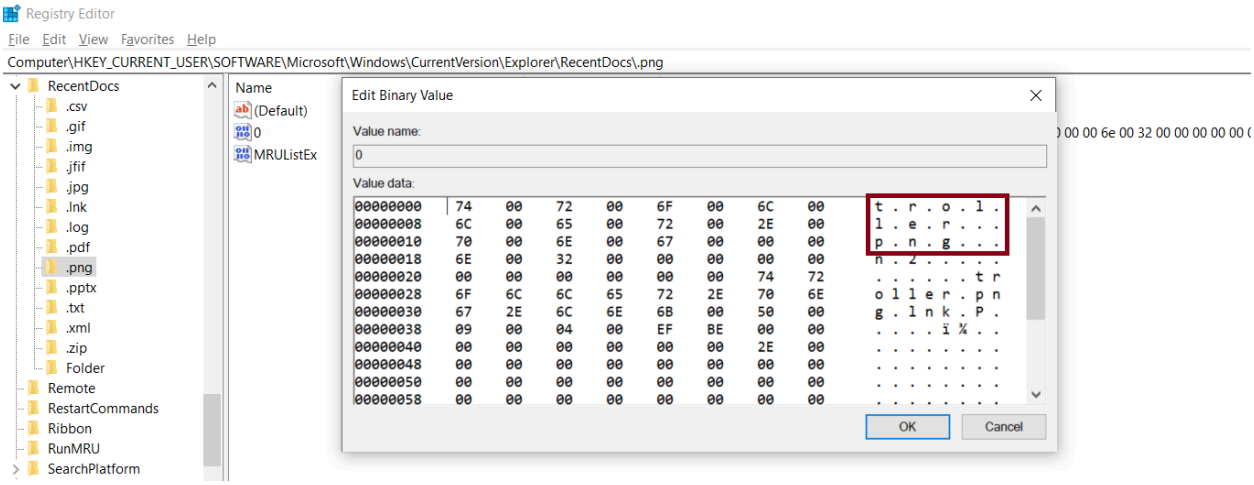

[72] https://medium.com/@boutnaru/the-windows-process-journey-explorer-exe-windows-explorer-9a96bc79e183
[73] https://digitalf0rensics.wordpress.com/2014/01/17/windows-registry-and-forensics-part2/
[74] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b
[75] https://forensic4cast.com/2019/03/the-recentdocs-key-in-windows-10/
[76] https://medium.com/@boutnaru/the-windows-process-journey-explorer-exe-windows-explorer-9a96bc79e183

# Recent Docs by Extension (Recently Used Documents by Extension)

The “RecentDocs”[77] registry key also has subkeys which are per extensions (such as “csv”/”gif”/”jpg”/”lnk”/”log”/”zip”/”xml”/”txt”/”pdf”/”ppt”/”pptx”/etc) and also for folders (on folders I am going to elaborate as part of a separate writeup) - as shown in the screenshot below.

Moreover, each of the extension's subkeys has its own “MRUListEx” with information regarding files from the same extension. Thus, we basically have duplicate data. For example if we open a “*.png” file it will appear both in “Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs” and in “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.png”[78] - as shown in the screenshot below.

Lastly, besides getting the list of files we can also get a quick view of types of files that have been accessed by a specific user (remember that the information is stored in HKCU). Alos, it is an easy way to understand what is the last file accessed based on a specific extension[79].



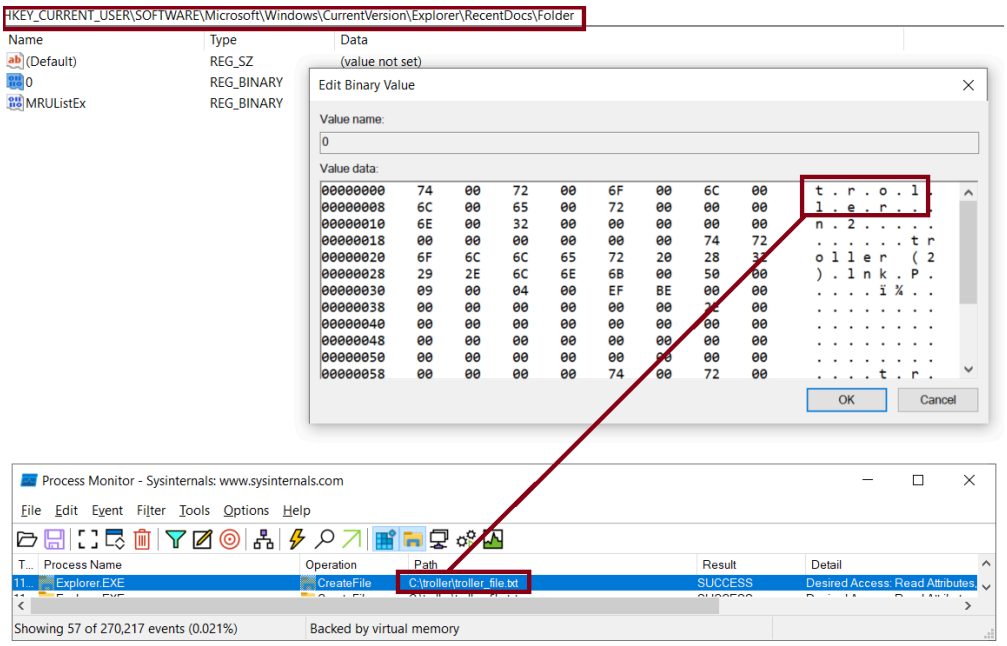

[77] https://medium.com/@boutnaru/the-windows-forensic-journey-recent-docs-recently-used-documents-a6d092d945ce
[78] https://forensic4cast.com/2019/03/the-recentdocs-key-in-windows-10/
[79] https://forensafe.com/blogs/recentdocs.html

# Folder of RecentDocs (Folder/s of Recently Used Documents)

The "RecentDocs"[80] registry key has subkeys for file extensions[81] and in conjunction with a subkey for folder/s. The key is located in the following "registry path"- "HKCU\SOFTWARE\Microsoft\Windows\Current Version\Explorer\RecentDocs\Folder".

Overall, the "Folder" subkey contains the folder of recently opened files. However, the folder is included without a drive letter and the part folder. Thus, opening a folder is not enough to trigger the collection of the information, we need to specifically open a file from the directory[82] - as shown in the screenshot below.

Lastly, we can use this indication regarding folder/s from which files were opened from even if it has been deleted since. Moreover, due to the fact the information is contained in HKCU we know which user has opened files for the specific folder/s.




[80] https://medium.com/@boutnaru/the-windows-forensic-journey-recent-docs-recently-used-documents-a6d092d945ce
[81] https://medium.com/@boutnaru/the-windows-forensic-journey-recent-docs-by-extension-recently-used-documents-by-extension-ff6deb94e880
[82] https://www.forensicfocus.com/articles/forensic-analysis-of-the-windows-registry/

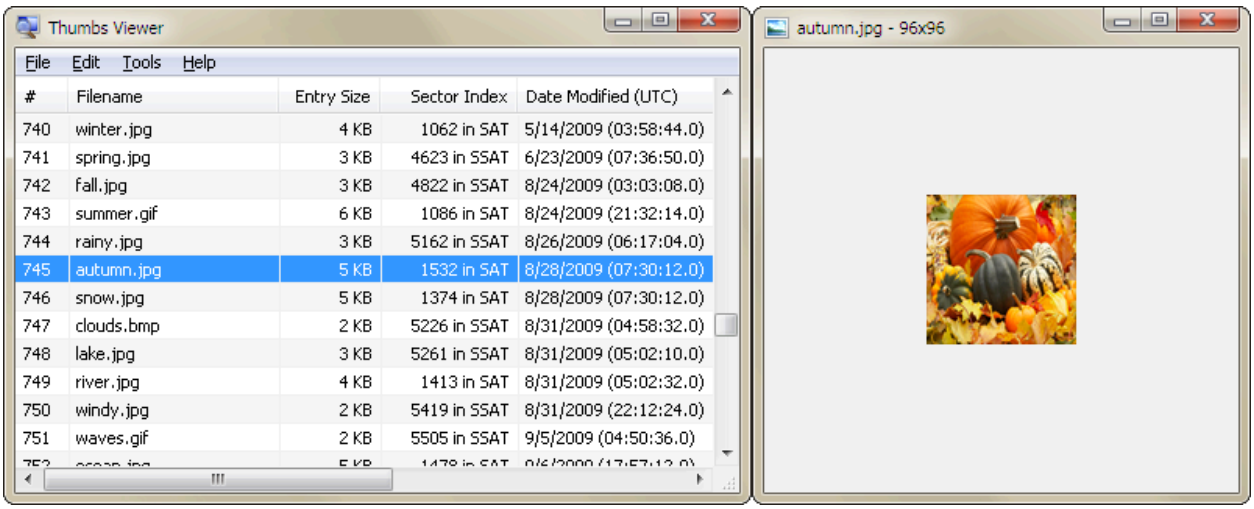
# Thumbs.db (Thumbnails Database)

Overall, thumbs.db files are hidden Windows system files generated to cache thumbnail images/first frame of videos/documents/HTML web pages/presentations. They represent the contents of image files when "Windows Explorer"[83] is set to thumbnails/filmstip views. It was created in order to display thumbnails faster due to the fact the operating system does not need to regenerate them every time the user accesses the directory. We can parse thumbs.db file using "Thumbs Viewer"[84] - as shown in the screenshot below.

Moreover, the thumbs.db artifacts are usually stored in the same directory as the folder whose thumbnails are cached. Also, thumbs.db data is stored in OLE compound file format. OLE is a binary format developed by Microsoft that works like a real file system[85].

By the way, from Windows Vista we have the "Thumb Cache" which is stored in a central location per user: "%userprofile%\AppData\Local\Microsoft\Windows\Explorer" (more on that in a future write). However, in case of network shares Windows Vista/7 store a thumbs.db file on the remote directory and not in the local cache[86].

Lastly, we can disable this feature using domain/local group policy by navigating to: User Configuration > Administrative Templates > Windows Components > then either Windows Explorer (Windows Vista/7) or File Explorer (Windows 8). Then we need to enable "Turn off the caching of thumbnails in hidden thumbs.db files"[87].




[83] https://medium.com/@boutnaru/the-windows-concept-journey-file-explorer-previously-windows-explorer-e48077b135a0
[84] https://thumbsviewer.github.io/
[85] https://forensafe.com/blogs/thumbdb.html
[86] https://en.wikipedia.org/wiki/Windows_thumbnail_cache
[87] https://www.sitepoint.com/switch-off-thumbs-db-in-windows/

# UserAssist

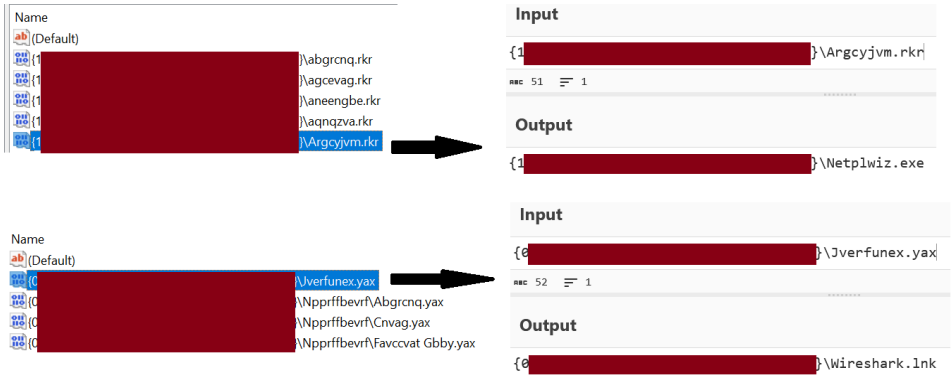
On a Windows based system, every GUI (Graphical User Interface) program which is launched from the desktop is track in the following registry key: "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{GUID}\Count"[88]. Due to the fact the key is under HKCU we can correlate the programs launched to a specific user.

Thus, under the "Count" subkey of the "{GUID}" subkey we can have a list of values which corresponds to executed programs. The names are encrypted using a simple letter substitution cipher called ROT13. It replaces each letter with the 13th letter after in the Latin alphabet[89] - as shown in the screenshots below.

Overall, The "CEBFF5CD-ACE2-4F4F-9178-9926F41749EA" GUID is used for listing of applications\files\links\other objects that have been accessed. Also, the "F4E57C4B-2036-45F0-A9AB-443BCFE33D9F" GUID is used for listing the shortcut links used to start programs[90]. Examples of both are shown in the screenshots below.

Moreover, as opposed to other forensics artifacts (like Prefetch) "UserAssist" includes information whether an application was run from a shortcut (LNK file - ) or by calling the executable directly[91]. Due to the encryption, if the name of the registry value ends with ".yax" it is a shortcut file (".lnk") and if it ends with ".rkr" it is an executable (".exe"). By the way, we can use "CyberChef" for decrypting RO13[92].

Lastly, "UserAssist" only tracks user interactions in "explorer.exe"[93]. This includes "Start->{Program}" but not "Start->Run->{Program}"[94].



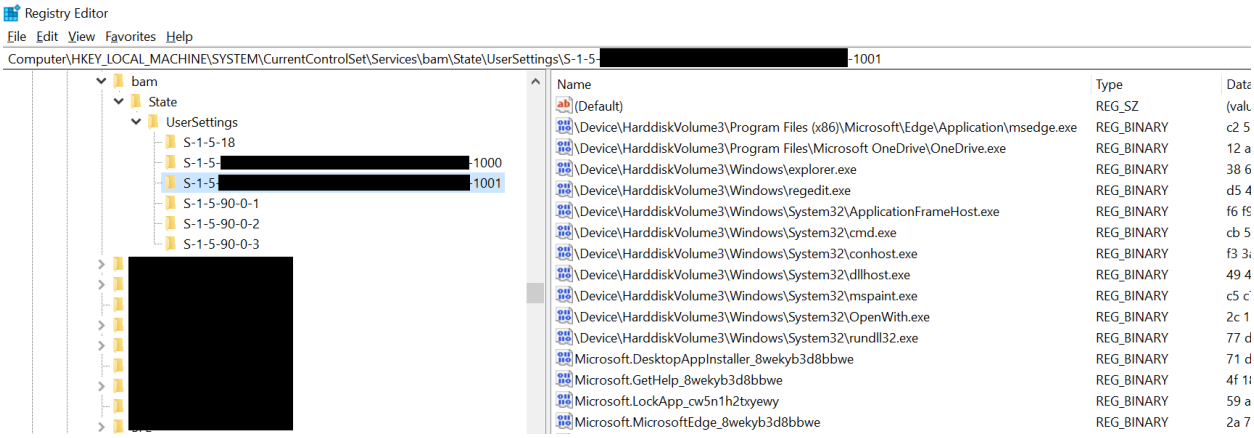

[88] https://andreafortuna.org/2018/05/23/forensic-artifacts-evidences-of-program-execution-on-windows-systems/
[89] https://en.wikipedia.org/wiki/ROT13
[90] https://windowsexplored.com/2012/02/06/a-quick-glance-at-the-userassist-key-in-windows/
[91] https://www.magnetforensics.com/blog/artifact-profile-userassist/
[92] https://gchq.github.io/CyberChef/
[93] https://medium.com/@boutnaru/the-windows-process-journey-explorer-exe-windows-explorer-9a96bc79e183
[94] https://github.com/marcurdy/dfir-toolset/blob/master/Windows_Artifacts.md

# BAM (Background Activity Moderator)

BAM (Background Activity Moderator) is used to control the activity of background applications. The entries in BAM are updated when Windows boots[95]. It is done using the BAM kernel driver ("%windir%\System32\drivers\bam.sys), which is digitally signed by Microsoft.

Moreover, BAM exists since Windows 10 fall creator update (version 1709). It stores binary data in the registry holding the execution of different programs/applications by users. The user attribution is based on the way in which the data is maintained in the registry under keys based on the SID[96] of the user executing the program/application[97]. At the beginning the data of BAM had been located at the following registry path "HKLM\SYSTEM\CurrentControlSet\Services\bam\UserSettings\{SID}", however it is now at "HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\{SID}"[98] - as shown in the screenshot below.

Lastly, BAM stores path to executables and the time of last execution in 64-bit little endian (aka Filetime). Also, we can use different tools to parse the BAM information like: BamParser and the different powershell scripts[99].




[95] https://www.digitalforensics.com/blog/news/new-windows-artifacts-background-activity-moderator-bam/
[96] https://medium.com/@boutnaru/windows-security-sid-security-identifier-d5a27567d4e5
[97] https://docs.velociraptor.app/docs/forensic/evidence_of_execution/
[98] https://github.com/Ektoplasma/BamParser
[99] https://github.com/kacos2000/Win10/blob/master/Bam/readme.md

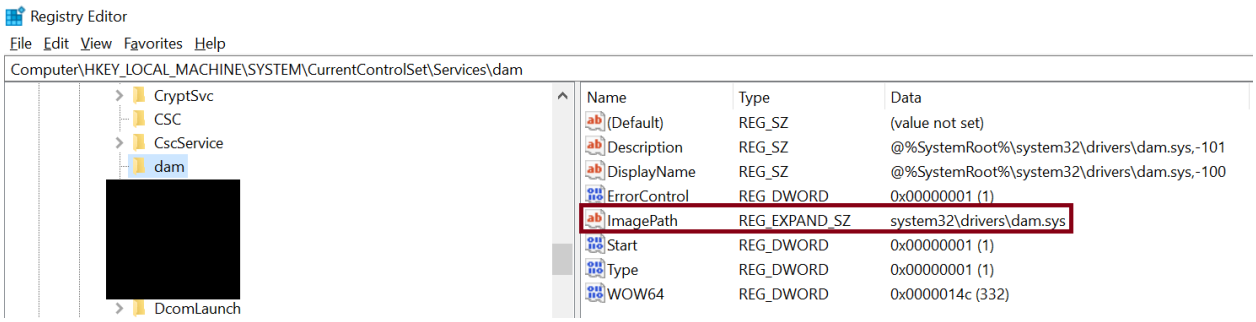
# DAM (Desktop Activity Moderator)

As with BAM[100], DAM (Desktop Activity Moderator) moderates desktop processes. It was created to ensure consistent and long battery life for devices that support "Connected Standby" (when the screen is off, but the device is still on).

Due to that, DAM is only populated with details of applications on Mobile/Tablet devices[101]. Thus, on normal PCs there is no data stored - as shown in the screenshot below. It is done using the BAM kernel driver ("%windir%\System32\drivers\dam.sys) - as shown in the screenshot below. The driver is digitally signed by Microsoft. DAM (as with BAM) is also updated when Windows boots.

Moreover, DAM entries (as also BAM entries) are only stored during a session, with events being cleared due to a reboot. Also, entries present more than 7 days are also cleared. It is important to know that executables launched from removal media are not recorded in DAM/BAM[102].

Lastly, the BAM entries are stored in "HKLM\System\CurrentControlSet\Services\dam\state\UserSettings\{SID}", where SID is unique per a local/domain user[103]. The information stored includes the last execution date/time and the full path to the executable[104].



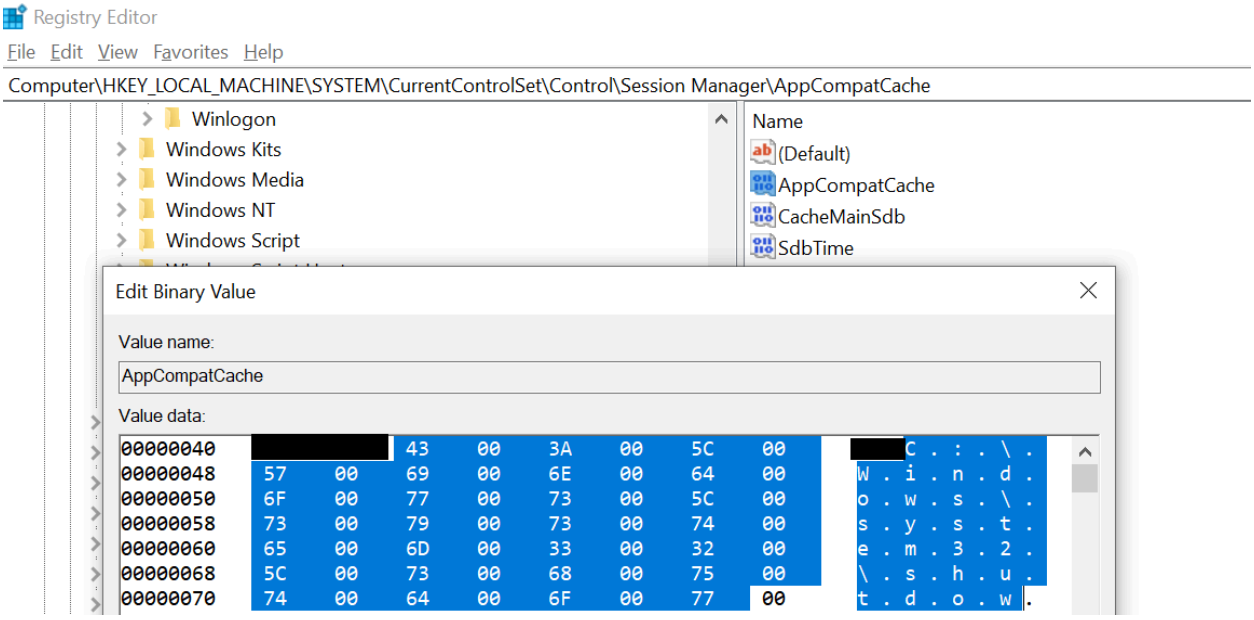

[100] https://medium.com/@boutnaru/the-windows-forensic-journey-bam-background-activity-moderator-729970b43e6c
[101] https://cellebrite.com/en/analyzing-program-execution-windows-artifacts/
[102] https://darkcybe.github.io/posts/DFIR_Evidence_of_Execution/
[103] https://medium.com/@boutnaru/windows-security-sid-security-identifier-d5a27567d4e5
[104] https://frsecure.com/blog/windows-forensics-execution/

# AppCompatCache (Application Compatibility Cache)

AppCompactCache (Application Compatibility Cache) is used in order to provide compatibility with old applications. By the way, it is also known as "ShimCache". AppCompactCache records the file path, file name and the last modification time and date, it can store information about local applications executed and also executables on UNC paths/removable media[105]. Since Windows 10 information about executables can be included even if they are not executed. Thus, we need to consider it as an indication of an executable being installed/present/accessed and not executed.

Moreover, the AppCompactCache/ShimCache has a limit of 1024 entries. We can find them in Windows 7/8/10/11 in the following registry location: "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache" - as shown in the screenshot below. For Windows XP it can be found in a different registry location: "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatibility"[106].

Lastly, we can use Eric Zimmerman's "AppCompatCacheParser" for parsing the "Application Compatibility Cache" and save the data to a CSV file[107]. It is important to know that AppCompactCache/SimCache entries are written to the registry on reboot/system shutdown. Thus, the data stored in the cache may not be the most up to date[108]. There is a Volatility plugin from Mandiant (it's repository was archived on Sep 15, 2021) which enables the extraction of the AppCompactCache/SimCache entries directly from a memory dump[109].



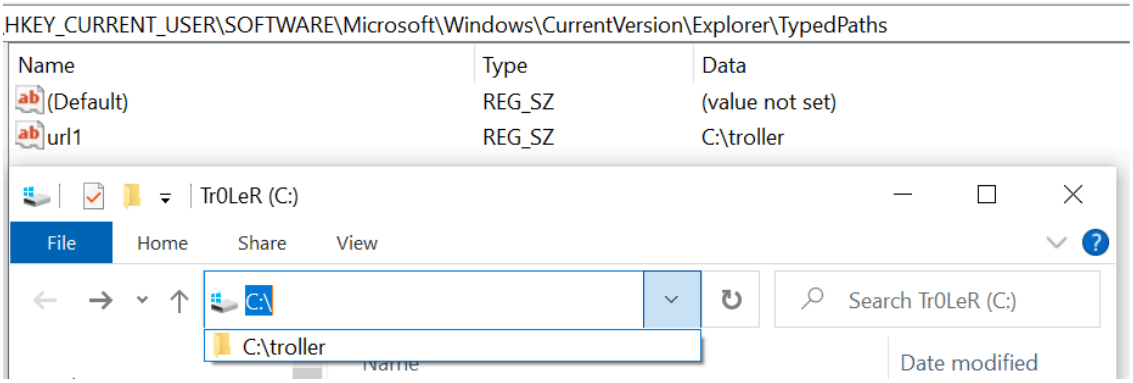

[105] https://forensafe.com/blogs/shimcache.html
[106] https://github.com/mandiant/ShimCacheParser/blob/master/README
[107] https://ericzimmerman.github.io/#!index.md
[108] https://www.thedfirspot.com/post/evidence-of-program-existence-shimcache
[109] https://github.com/mandiant/Volatility-Plugins/blob/master/shimcachemem/shimcachemem.py

# TypedPaths (Addresses Typed in File Explorer)

"Typed Paths" is a Windows registry key which tracks the last 25 paths that have been entered into the path bar of "File Explorer"[110]. In order for the paths to appear in the "TypedPaths" registry key we need to close the "File Explorer" window for the data to be committed[111].

Overall, the registry path and the "Typed Paths" registry key is: "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths" - as shown in the screenshot below. Using this information we can try and recreate a timeline of the user's activities[112].

Lastly, the paths stored in the registry key can be local or remote (like when using UNC paths while accessing a SMB share or URLs). Due to that we can find information like: installed applications, network settings (based on DNS suffixes/IPs and more), user activity and more[113].



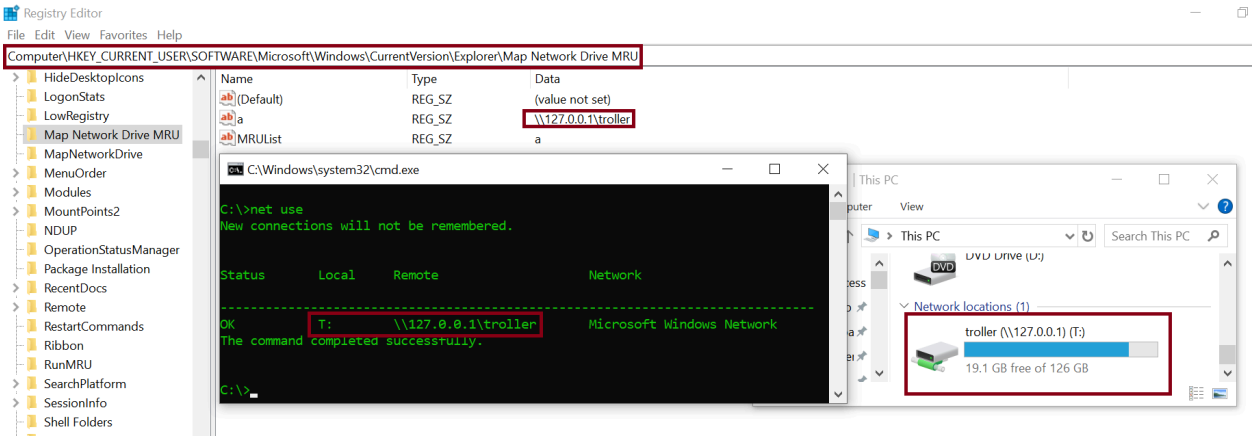

[110] https://medium.com/@boutnaru/the-windows-concept-journey-file-explorer-previously-windows-explorer-e48077b135a0
[111] https://forensafe.com/blogs/typedpaths.html
[112] https://www.3fforensics.com/forensics/typed-paths.html
[113] https://www.4n6post.com/2023/02/registry-typedpath.html

# Map Network Drive MRU (Recently Mapped Network Drives)

“Map Network Drive MRU” is a Windows registry key which stores information about the recently mapped network drives. A network drive is basically a way of mapping a shared folder[114] to a device drive - as shown in the screenshot below. The location of the key is: “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Map Network Drive MRU”. As we can see it contains information per logged on user.

Overall, a network drive can be mapped using the UI of “File Explorer”[115] or by using the “net.exe”[116] command, which can also be used to view mapped drives - as shown in the screenshot below.

Lastly, due to the fact registry keys have a last write timestamp we can know when the mapping was created. Also, from the location of the share we can also get information such as IPs/DNS names/NetBIOS names used by the user. We can get this information from a live system, remotely (in case the “Remote Registry” service and it is accessible over the network) or by parsing “NTUSER.DAT” files.



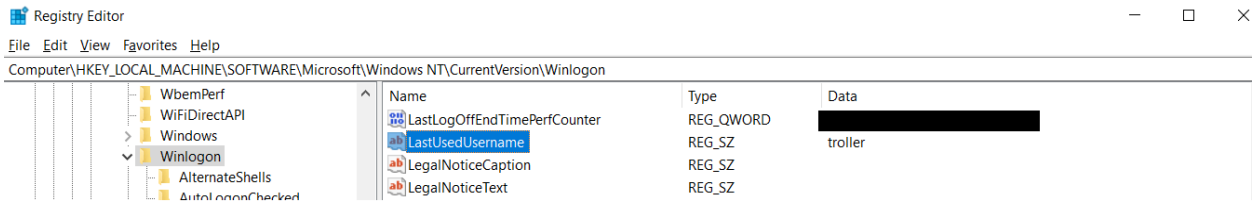

[114] https://medium.com/@boutnaru/the-windows-concepts-journey-windows-shares-8f9b60b8efd1
[115] https://medium.com/@boutnaru/the-windows-concept-journey-file-explorer-previously-windows-explorer-e48077b135a0
[116] https://medium.com/@boutnaru/the-windows-process-journey-net-exe-net-command-91e4964f20b8

# LastUsedUsername (Username of the Last Logged On User to the System)

"LastUsedUsername" is a value name in the registry[117] that holds the username of the last logged on user to the system. The full location in the registry is "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon". Due to the fact this information is global to the system it is part of the HKLM hive - as shown in the screenshot below.

Thus, we can read this value in order to know what was the last user which logged on to the system (even if the relevant event log entries have been deleted). We can also correlate that with MAC (Modified/Accessed/Create) times of folders in the user's profile directory[118].

Lastly, we can access this information by investigating a live machine, an offline SYSTEM hive or from a remote machine (in case the "Remote Registry" service is running and the machine is accessible using SMB/MS-RPC).

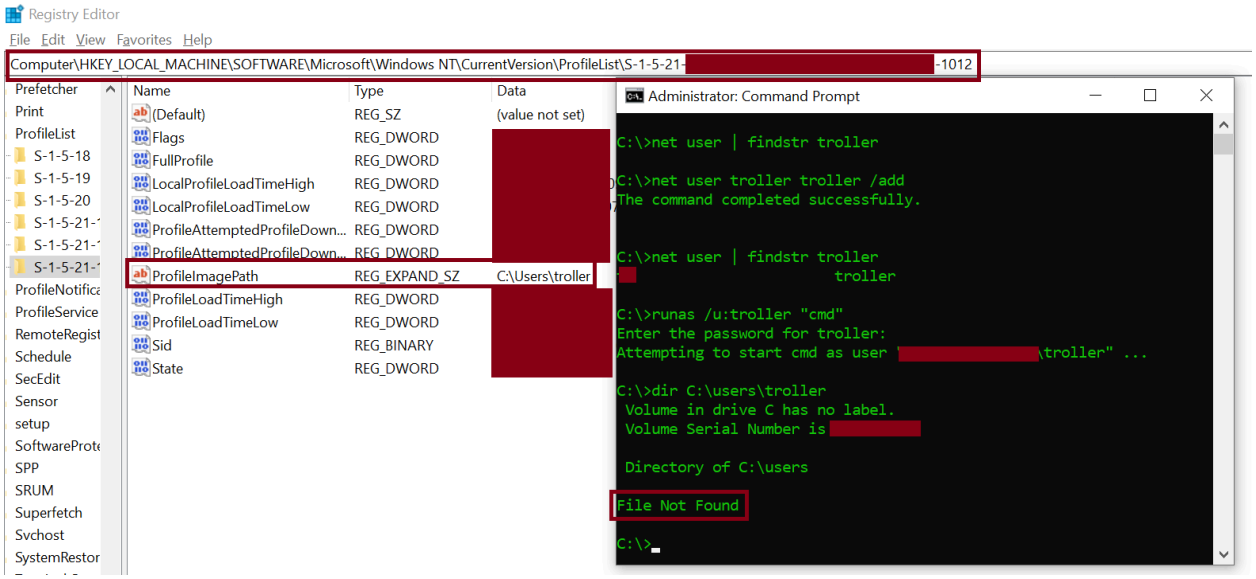



[117] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[118] https://medium.com/@boutnaru/the-windows-concept-journey-user-profile-555c23cc6a7e

# ProfileList (User's Profiles List)

"ProfileList" is a registry key[119] that holds information about user profiles[120] created on a specific Windows machine.
Moreover, the full location in the registry is "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList". Due to the fact this information is global to the system it is part of the HKLM hive[121] - as shown in the screenshots below.

Thus, we can use it to identify user accounts that have logged on even if their user profile directory had been deleted (in case of a domain account). Also, in case of a local user account even if the user has been deleted (but at least one logon was performed) we can know that there was such a user - as shown in the screenshots below. Finally, we can extract from the "ProfileList" different data points like the SID (Security Identifier) of the user account, the profile path and the last update time[122].

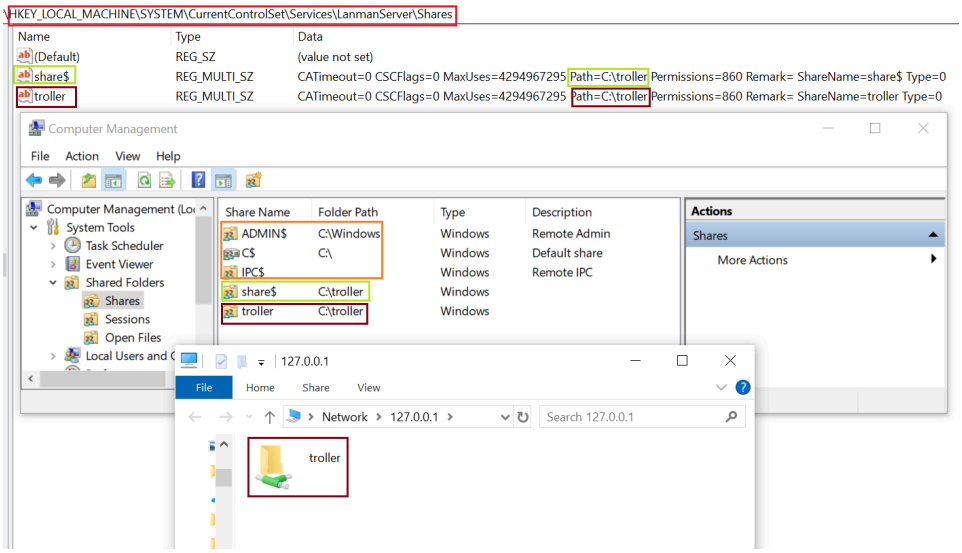



[119] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[120] https://medium.com/@boutnaru/the-windows-concept-journey-user-profile-555c23cc6a7e
[121] https://www.firatboyan.com/en/local-user-profiles-in-windows-10.aspx
[122] https://forensafe.com/blogs/profileslist.html

# Shared Folders (Windows Shares/Network Shares)

The goal of a share folder/Windows share is to expose a folder over the network. This allows users to access files which reside remotely on a different Windows machine[123].Moreover, as opposed to the folder them self in which the permissions is saved as part of the NTFS data structures, the permissions of network shares are saved in the registry[124].

Lastly, the name and the folder's path (shared by the network share) are also stored in the registry as part of the" LanmanServer" service (aka "Server" service) configuration. The full location is "HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Shares"[125]. We can see all existing shares including the hidden ones (those ending with "$"), except the default ones - as shown in the screenshot below.



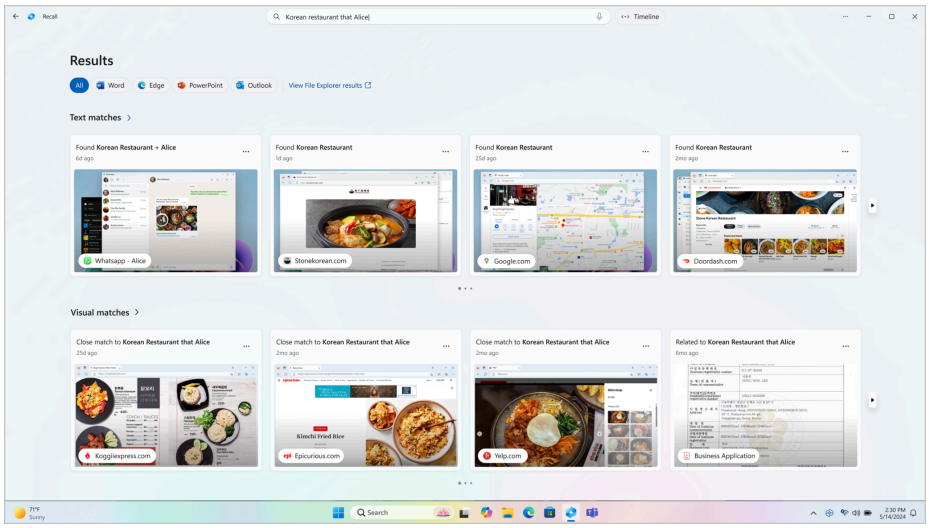

[123] https://medium.com/@boutnaru/the-windows-concepts-journey-windows-shares-8f9b60b8efd1
[124] https://medium.com/@boutnaru/the-windows-security-journey-share-permissions-network-shares-2a5beb7a8f96
[125] https://learn.microsoft.com/en-us/troubleshoot/windows-client/networking/saving-restoring-existing-windows-shares

# Windows Recall

The purpose of “Windows Recall” is to allow users to retrace things that they have done on a specific Windows system. By using recall the operating system provides an explorable timeline of the user actions. Thus, we just need to describe how we remember what we want to retrace and Recall will take us to that point in time- as shown in the screenshot below. This is done by taking a snapshot (stored locally) every 5 seconds (while the screen content is changed)[126].

Moreover, “Windows Recall” has the following minimum system requirements: 8 logical processors, 16 GB RAM and at least 50 GB of space for enabling recall (256 GB is recommended). The last requirement is “Copilot + PC”, that is a new class of Windows 11 system that is powered by a turbocharged NPU (neural processing unit) – a computer chip for AI-intensive processes[127].

Lastly, we can control which applications we exclude from recall (think about banking apps\websites). Of course it is relevant only for supported browsers (in regards to websites), we can add such filters in “Windows Settings > Privacy & Security > Recall & Snapshots” and click on “Add website” or “Add App” defending on what we want to exclude[128]. We can access the recall feature using “WinKey+J” key combination or by a dedicated key if we have it in our keyboard.



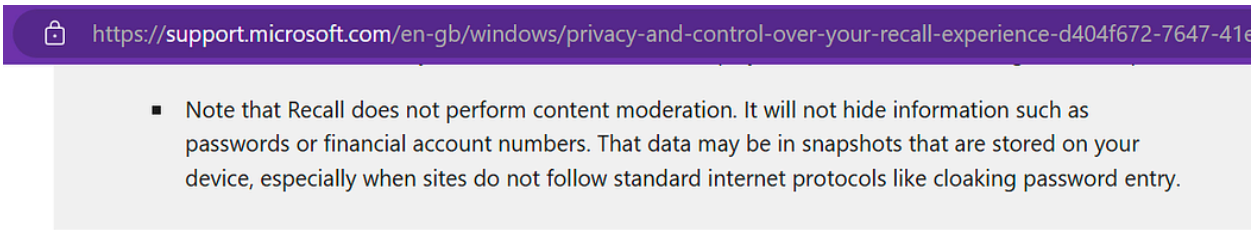

[126] https://support.microsoft.com/en-us/windows/retrace-your-steps-with-recall-aa03f8a0-a78b-4b3e-b0a1-2eb8ac48701c
[127] https://www.microsoft.com/en-gb/windows/copilot-plus-pcs#faq1
[128] https://pureinfotech.com/exclude-apps-websites-recall-windows-11/

# Windows Recall's Artifacts

In general, we can use "Windows Recall" to retrace things that they have done on a specific Windows system. Those points in time are shown as a time that can be accessed by the user. For privacy reasons we can disable "Recall" if we want. Also, it is not enabled by default[129].

The information we can extract from recall can include: indication of a process execution, indication of working with specific folders, timestamp for activities and more. The recall data is saved per-user in the user's profile directory "C:\Users\%username%\AppData\Local\CoreAIPlatform.00\UKP\{GUID}\". The relevant artifacts are: "ImageStore" directory (stores the screenshot taken in JPEG format), "ukg.db" (SQLite database which contains ULRs and OCRed text) and "SemanticTextStore.sidb" & "SemanticImageStore.sidb" which are DiskANN graph database[130] - more information about each in future writeups.

Lastly, "DiskANN" is a suite of scalable, accurate and cost-effective approximate nearest neighbor search algorithms for large-scale vector search that support real-time changes and simple filters maintained by Microsoft[131]. By the way, the artifacts can contain sensitive data due to that fact "Recall" does not perform any content moderation - as explained in the screenshot below[132].



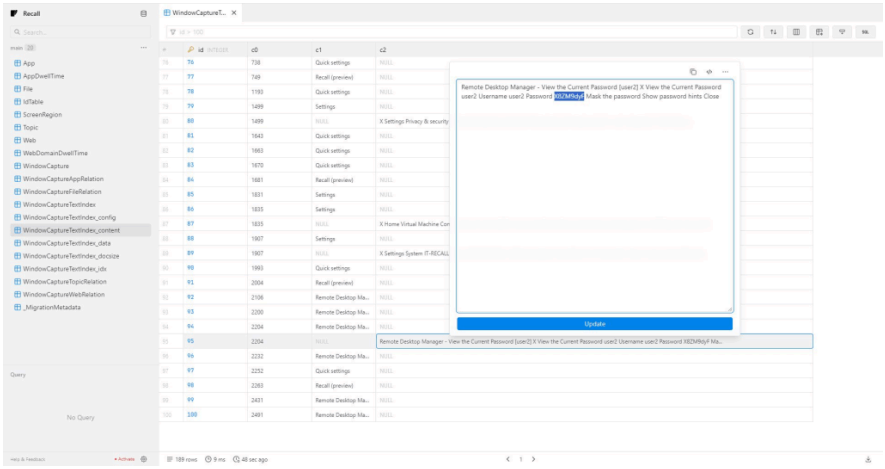

[129] https://medium.com/@boutnaru/the-windows-forensic-journey-windows-recall-2e31d1844767
[130] https://cybercx.com.au/blog/forensic-applications-of-microsoft-recall/
[131] https://github.com/microsoft/DiskANN
[132] https://privsecure.com.au/windows-recall-spyware/

# ukg.db (Windows Recall)

The “ukg.db” is part of the “Window Recall” feature. It is an SQLite database which is used for storing information as text and mapping it to the screenshots that were taken[133]. Just as a reminder to enable “Windows Recall” we need a Copilot+ PCs[134].

Overall, the “ukg.db” which stores all the information that recall stores (beside the screenshots) like the name of the app and the content that was extracted from the screenshot/snapshots[135]. Also, there are different open source tools that can parse/extract information from “ukg.db” such as “TotalRecall”[136].

Lastly, the database has 20 tables: “App”, “AppDwellTime”, “File”, “idTable”, “ScreenRegion”, “Topic”, “Web”, “WebDomainDwellTime”, “WindowCapture”, “WindowCaptureAppRelation”, “WindowCaptureFileRelation”, “WindowCaptureTextIndex”, “WindowCaptureTextIndex_config”, “WindowCaptureTextIndex_content”, “WindowCaptureTextIndex_data”, “WindowCaptureTextIndex_docsize”, “WindowCaptureTextIndex_idx”, “WindowCaptureTopicRelation”, “WindowCaptureWebRelation” and “_MigrationMetadata” - as shown in the screenshot below[137].



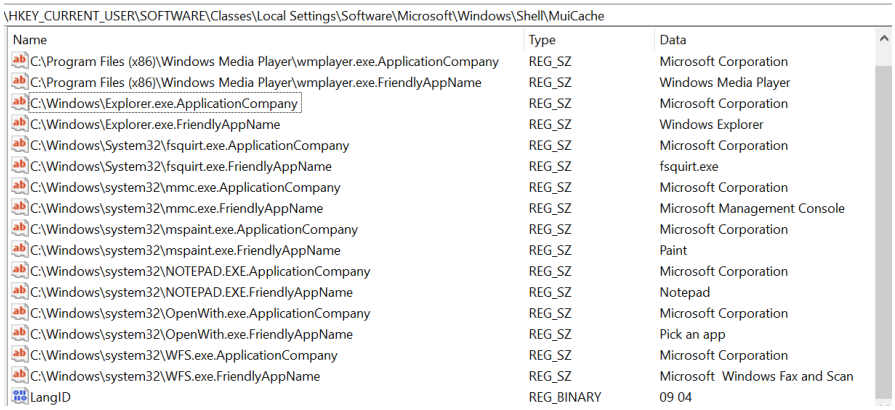
[133] https://medium.com/@boutnaru/the-windows-forensic-journey-windows-recall-2e31d1844767
[134] https://www.microsoft.com/en-us/windows/copilot-plus-pcs?r=1#shop
[135] https://pureinfotech.com/access-recall-ai-data-locally-stored-windows-11/
[136] https://github.com/xaitax/TotalRecall
[137] https://msandbu.org/how-does-windows-recall-work/

# MUICache (Multilingual User Interface Cache)

MUI (Multilingual User Interface Cache) is a technology that is used for enabling multilingual user experiences without the need of changing the binaries on the operating system[138]. Due to the fact it can be used by applications "MUICache" can provide information about installed applications and execution per-user account. This information can persist even if the application is deleted/removed. It is important to understand that it won't contain the entire list of executed processes on the system.

Overall, since Windows Vista the location of the "MUICache" key is: "HKEY_CURRENT_USER\Software\Classes\Local\Settings\Software\Microsoft\Windows\Shell\MuiCache", thus it is stored as part of the "UsrClass.dat" file[139]. Prior to Vista it was "HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache". Since Windows Vista "MUICache" is located at "HKCU\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache"[140] - as shown in the screenshot below.

Lastly, "MUICache" contains the following forensic artifacts: the file name of the executed binary, file path of the binary and additional data about the file[141] - as shown in the screenshot below.



\HKEY_CURRENT_USER\SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache

| Name | Type | Data |
|---|---|---|
| C:\Program Files (x86)\Windows Media Player\wmplayer.exe.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Program Files (x86)\Windows Media Player\wmplayer.exe.FriendlyAppName | REG_SZ | Windows Media Player |
| C:\Windows\Explorer.exe.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Windows\Explorer.exe.FriendlyAppName | REG_SZ | Windows Explorer |
| C:\Windows\System32\fsquirt.exe.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Windows\System32\fsquirt.exe.FriendlyAppName | REG_SZ | fsquirt.exe |
| C:\Windows\system32\mmc.exe.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Windows\system32\mmc.exe.FriendlyAppName | REG_SZ | Microsoft Management Console |
| C:\Windows\system32\mspaint.exe.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Windows\system32\mspaint.exe.FriendlyAppName | REG_SZ | Paint |
| C:\Windows\system32\NOTEPAD.EXE.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Windows\system32\NOTEPAD.EXE.FriendlyAppName | REG_SZ | Notepad |
| C:\Windows\system32\OpenWith.exe.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Windows\system32\OpenWith.exe.FriendlyAppName | REG_SZ | Pick an app |
| C:\Windows\system32\WFS.exe.ApplicationCompany | REG_SZ | Microsoft Corporation |
| C:\Windows\system32\WFS.exe.FriendlyAppName | REG_SZ | Microsoft Windows Fax and Scan |
| LangID | REG_BINARY | 09 04 |

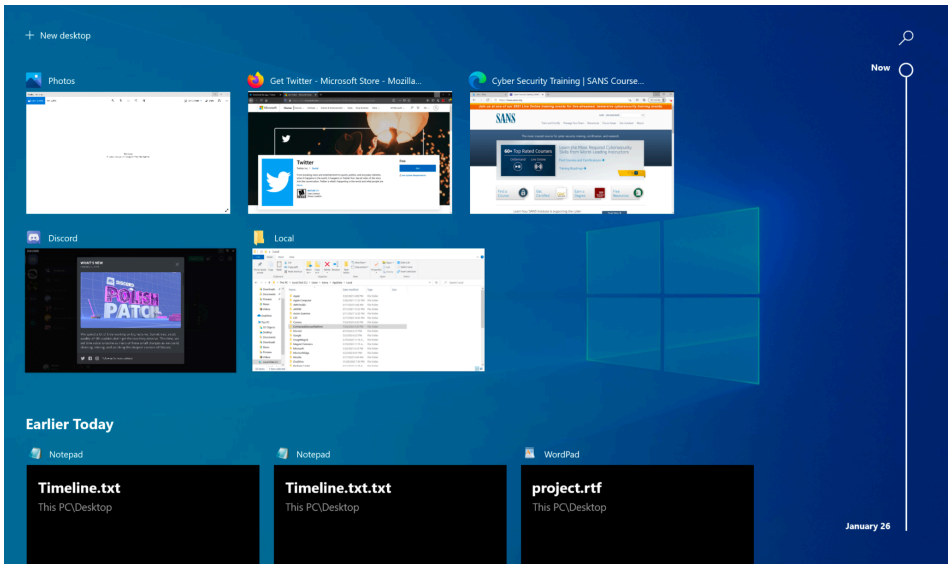
[138] https://medium.com/@boutnaru/the-windows-concept-journey-multilingual-user-interface-mui-c225998d9262
[139] https://medium.com/@boutnaru/the-windows-concept-journey-b8b2a4476f47
[140] https://openmuifile.com/muicache.html
[141] https://forensafe.com/blogs/muicache.html

# Windows Timeline

The “Windows Timeline” feature was introduced as part of Windows 10 (version 1803). By using these features a user can checkout current running applications and look back in a timeline on activities done in the past. Examples of such activities are: opened applications/documents/images/videos/websites/etc - as shown in the screenshot below[142].

Overall, we can access the “Windows Timeline” using “WinKey+Tab” or by clicking the “Task View” icon located in the task bar. Also, this feature can be used in order to synchronize activities across different devices[143]. By the way, this feature is also sometimes called “Activity History”.

Lastly, by default the data is stored in “Windows Timeline” for 3-4 days. In case we logon with a Microsoft account the data is stored for up to 30 days[144]. One of the drawbacks is that we can’t limit “Windows Timeline” to stop monitoring a specific application as we can do with “Windows Recall”[145].



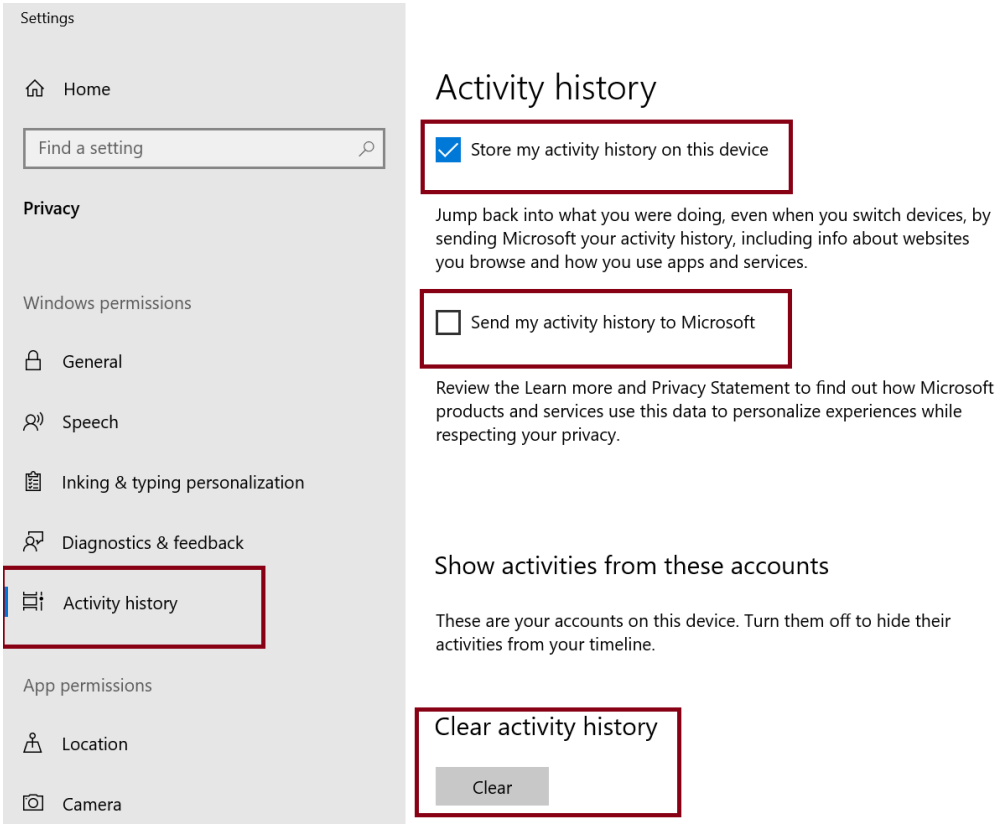

[142] https://forensafe.com/blogs/windowstimeline.html
[143] https://istrosec.com/blog/windows-10-timeline/
[144] https://www.digitalcitizen.life/what-is-timeline-how-use-resume-past-activities/
[145] https://medium.com/@boutnaru/the-windows-forensic-journey-windows-recall-2e31d1844767

# Activity History (Jump Back To What You Were Doing)

The goal of "Activity History" is to keep track of the thing the user is doing on a specific device (applications/services in use, files opened, website browsed). This information can be used to personalize the experience while using Windows. Examples for that are ordering the user activities based on duration of use or anticipating the user needs based on their activities[146].

By default, the "Activity History" is stored locally, however if we give permissions and logon with a school/work account Windows can send the information collected to Microsoft - as shown in the settings' screen in the screenshot below (Settings->Privacy->Activity History). By sending the information to Microsoft the user can jump back into activities that have been done in different devices (this is not configured by default).

Lastly, "Activity History" is used by different Windows features (Timeline and Microsoft Edge - more on that in future writeups). Also, Beside disabling the ability to store "Activity History" we can also "Clear Activity History" - as shown in the screenshot below.



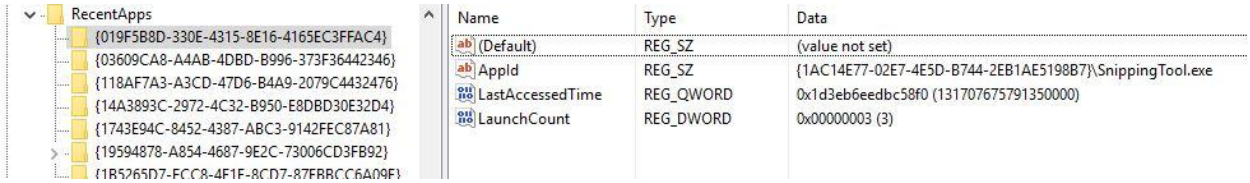

[146] https://support.microsoft.com/en-us/windows/-windows-activity-history-and-your-privacy-2b279964-44ec-8c2f-e0c2-6779b07d2cbd

# RecentApps

RecentApps is a feature relevant since Windows 10 which logs execution of GUI programs. It is saved per local/domain user in the following registry location: "HKCU\Software\Microsoft\Windows\CurrentVersion\Search\RecentApps"[147]. It seems that Windows 10 tracked program execution using "Recent Apps" versions 1607-1709[148].

Moreover, every sub-key of "RecentApps" contains information about a specific application (identified by GUIDs). The values included are: "AppID" (name of the application), "LastAccessTime" (the last execution time in UTC format) and "Launch Count" (number of times the application was executed) - as shown in the screenshot below[149].

Lastly, the information is saved per user as part of its user profile. Thus, we can parse the information from a "NTUSER.DAT" file[150]. Also, in specific cases some GUID subkeys can have their own additional subkeys which correspond to particular files accessed by the application[151].



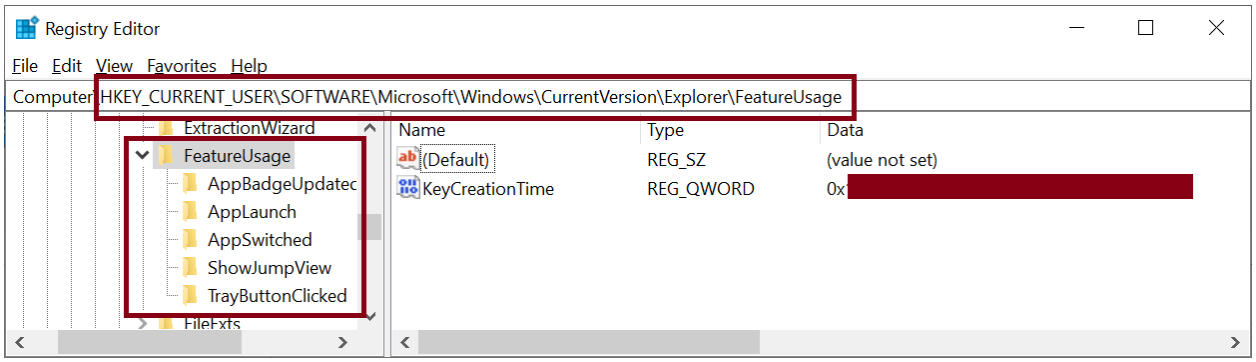

[147] https://andreafortuna.org/2018/05/23/forensic-artifacts-evidences-of-program-execution-on-windows-systems/
[148] https://darkcybe.github.io/posts/DFIR_Evidence_of_Execution/
[149] https://andreafortuna.org/2018/05/23/forensic-artifacts-evidences-of-program-execution-on-windows-systems/
[150] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[151] https://df-stream.com/2017/10/recentapps/

# FeatureUsage

In general, “FeatureUsage” is a registry key which is stored as part of the user’s profile. This means that the information is stored for each in the NTUSER.DAT file[152]. The location of the registry key is: “HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage” - as shown below. It is created only when a user performs an interactive login which can be locally or RDP based[153].

Overall, the “FeatureUsage” key has a QWORD value called “KeyCreationTime” which can give us the first time the user performed an interactive logon - as shown in the screenshot below. The data is stored in the following format: “number of 100-nanosecond intervals that have passed since January 1, 1601 UTC”. We can convert it from “Window Filetime” to “Unix Timestamp” and from that to a valid datetime format which is human readable[154]. This can be done using “CyberChef”[155].

Lastly, the “FeatureUsage” registry key has five sub-keys: “AppBadgeUpdated”, “AppLaunch”, “AppSwitched”, “ShowJumpView” and “TrayButtonClicked” - as shown in the screenshot below. Each of those sub-keys provides information about executables that were launched on the system - more information about each one of them in future writeups.



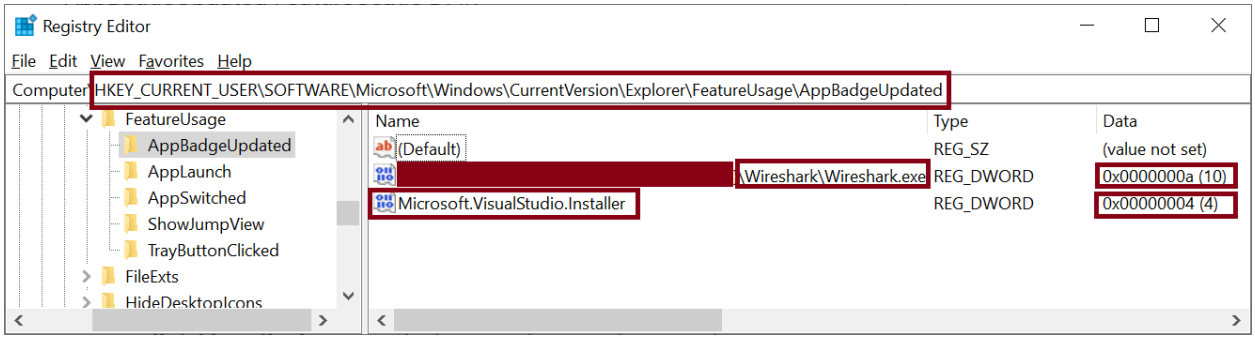

[152] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[153] https://www.group-ib.com/blog/featureusage/
[154] https://www.crowdstrike.com/blog/how-to-employ-featureusage-for-windows-10-taskbar-forensics/
[155] https://gchq.github.io/CyberChef/

# AppBadgeUpdated

The "AppBadgeUpdated" is a registry subkey of "FeatureUsage"[156]. By inspecting "AppBadgeUpdated" we can understand the number of times an application had its icon badge updated (on the taskbar). Think for example of an email application that modifies the icon due to unread emails/notifications[157]. The location of the registry key is as follows: "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\AppBadgeUpdated".

Thus, using this subkey we can try and understand how a specific application was used (think for example on an IM application). We can retrieve from the "AppBadgeUpdated" sub-key the name of the application and/or a path to the application executable[158] - as shown in the screenshot below.

Lasty, it is important to understand that even if an application is deleted/uninstalled information about it can be found as part of the "AppBadgeUpdated" artifacts. This is due to it not being removed when the original application is removed. Also, we can read the information directly from a NTUSER.DAT file[159].



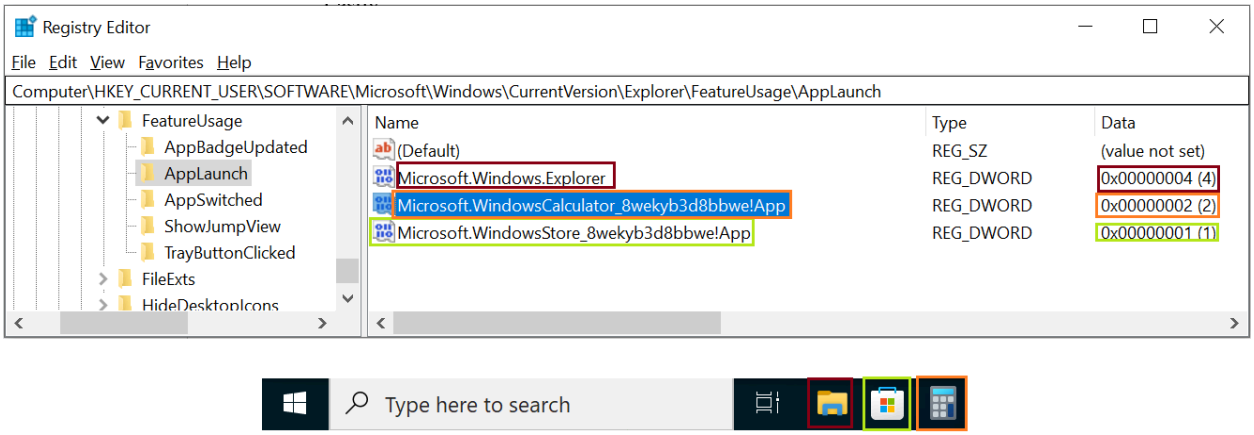

[156] https://medium.com/@boutnaru/the-windows-forensic-journey-featureusage-aed8f14c84ab
[157] https://www.crowdstrike.com/blog/how-to-employ-featureusage-for-windows-10-taskbar-forensics/
[158] https://www.jaiminton.com/cheatsheet/DFIR/#recent-execution-of-programs-gui
[159] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349

# AppLaunch

The "AppLaunch" is a registry subkey of "FeatureUsage"[160]. The location of the registry key is: "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\AppLaunch". Also, we can read the information directly from a NTUSER.DAT file[161].

Overall, "AppLaunch" records the launches of applications which are pinned to the taskbar. Also, each time a pinned application is started from the taskbar the counter of "number of execution" is updated - as shown in the screenshot below. The list of launched applications logged in "AppLaunch" is a subset of all executed applications due to the fact not all of them are pinned to the taskbar[162].

Lastly, it is important to understand that if an application is launched not from the taskbar the information is not saved as part of "AppLaunch", like when running apps using "WinKey+R"[163].



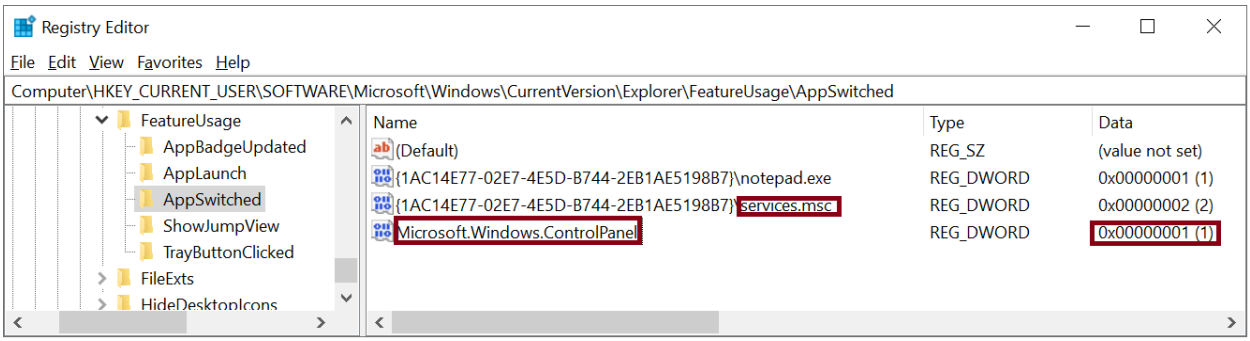

[160] https://medium.com/@boutnaru/the-windows-forensic-journey-featureusage-aed8f14c84ab
[161] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[162] https://www.group-ib.com/blog/featureusage/
[163] https://medium.com/@boutnaru/the-windows-foreniscs-journey-run-mru-run-dialog-box-most-recently-used-57375a02d724

# AppSwitched

The "AppSwitched" is a registry subkey of "AppSwitched"[164]. The location of the registry key is "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\AppSwitched ". Also, we can read the information directly from a NTUSER.DAT file[165].

Overall, "AppSwitched" logs information about an application when a user "Left Clicks" on the application (and a counter of the number of times it has happened) in the taskbar in order to switch to it[166] - as shown in the screenshot below. Also, when running a control panel applet[167] like "ncpa.cpl", the "AppSwitched" list will contain "Microsoft.Windows.ControlPanel" - as shown below.

Lastly, in case we use a "*.msc" file although it is loaded by "mmc.exe"[168] the name of "*.msc" appears in the "AppSwitched" list and not "mmc.exe" - as shown in the screenshot below.



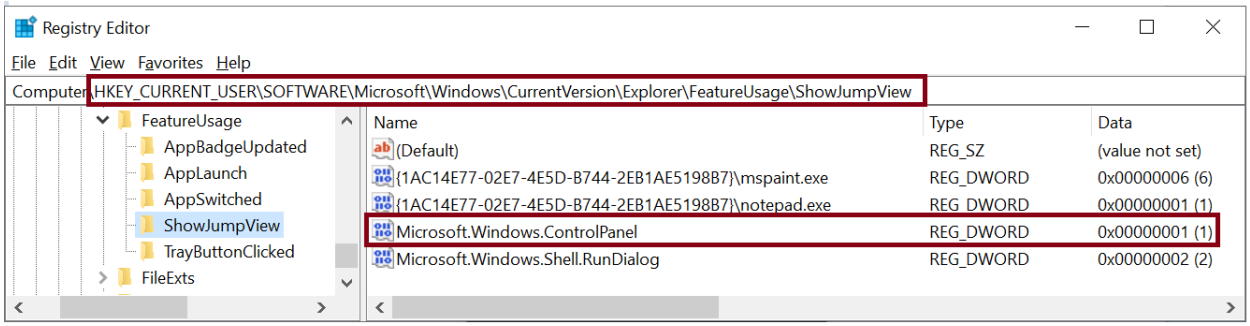

[164] https://medium.com/@boutnaru/the-windows-forensic-journey-featureusage-aed8f14c84ab
[165] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[166] https://www.group-ib.com/blog/featureusage/
[167] https://medium.com/@boutnaru/the-windows-concept-journey-control-panel-34bf84ca7ff0
[168] https://medium.com/@boutnaru/the-windows-process-journey-mmc-exe-microsoft-management-console-a584afe66d86

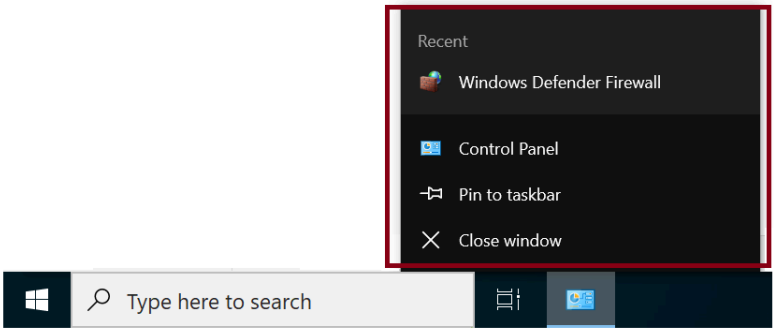
# ShowJumpView

The "ShowJumpView" is a registry subkey of "FeatureUsage"[169]. The location in the registry is: "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\ShowJumpView". Due to that fact "ShowJumpView" is saved per user (as part of its user profile) , we can parse the information directly from a NTUSER.DAT file[170].

Moreover, the "ShowJumpView" artifact records information about the number of times the user has right clicked on an application appearing on the taskbar. Thus, we can get a general understanding on the usage frequency for an application[171].

Lastly, when "Left Clicking" on an app icon located on the taskbar a "jump list"[172] is shown - as shown in the screenshot below. It is important to understand that "Jump Lists" are also supported in the "Start Menu". However, seeing the information in the start menu does not have any effect on "ShowJumpView" artifact (no data is recorded in that case).




[169] https://medium.com/@boutnaru/the-windows-forensic-journey-featureusage-aed8f14c84ab
[170] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[171] https://www.group-ib.com/blog/featureusage/
[172] https://medium.com/@boutnaru/the-windows-concept-journey-jump-list-8be7eb2c66bd

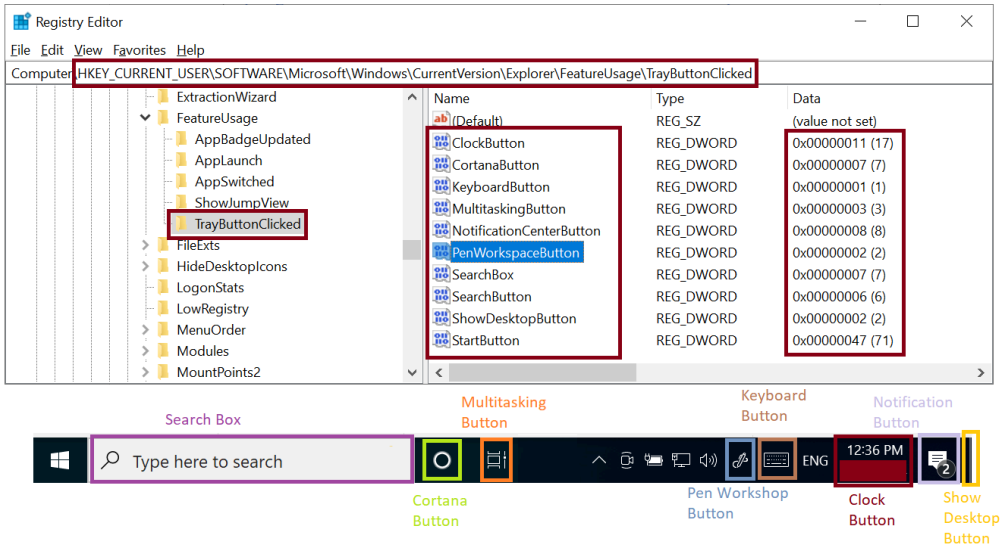
# TrayButtonClicked

The "TrayButtonClicked" is a registry subkey of "FeatureUsage"[173], located at: "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\TrayButtonClicked". Due to that fact "TrayButtonClicked" is saved per user (as part of its user profile) , we can parse the information directly from a NTUSER.DAT file[174].

Overall, "TrayButtonClicked" has different available values such as: "ClockButton", "MultitaskingButton", "NotificationCenterButton", "SearchBox", "SearchButton", "PenWorkspaceButton", "KeyboardButton", "CortanaButton" "StartButton" and "ShowDesktopButton" - as shown in the screenshot below. Each of those has a data which contains the number of times a specific area in the taskbar has been clicked by the user[175].

Thus, "ClockButton" is relevant for the number of times the Clock button was clicked. "MultitaskingButton" represents the clicking counter of the multitasking button (the one that has the "Task View" appearing when hovering on it). "NotificationCenterButton" holds the counter of clicks regarding the notification center. "SearchBox" which is relevant to the search text box, while the "SearchButton" is in case the button is shown in the taskbar instead. "ShowDesktopButton" which holds the number of times the show desktop button was clicked[176].

Lastly, "KeyboardButton" is in case the user pressed the touch keyboard button. "CortanaButton" holds the counter for pressing the "talk to cortana" button. "PenWorkspaceButton" is updated when the "windows ink workspace" is pressed on the taskbar. The different buttons are shown and marked in the screenshot below.



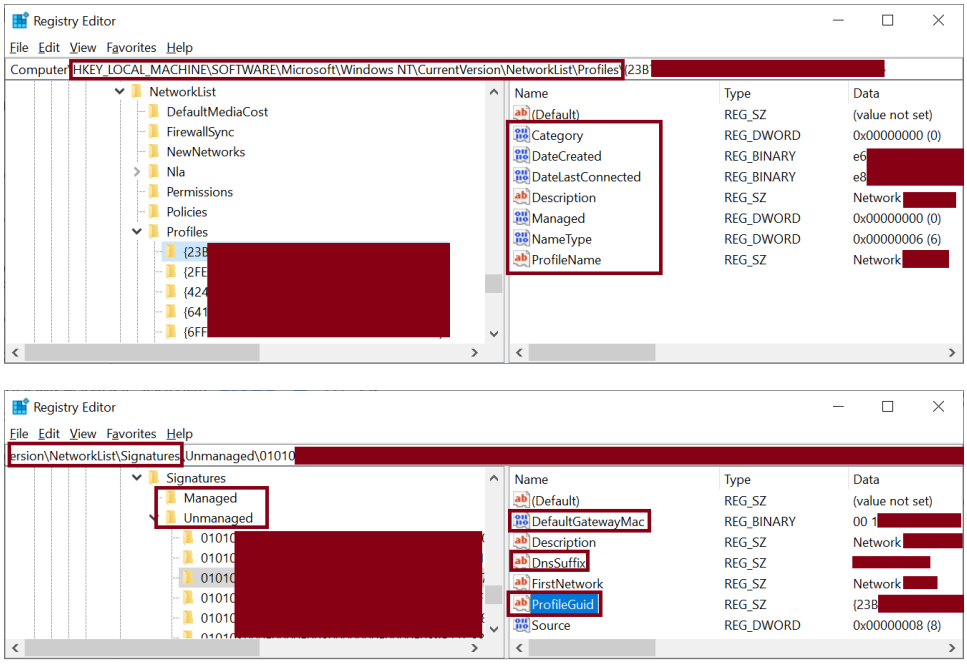

[173] https://medium.com/@boutnaru/the-windows-forensic-journey-featureusage-aed8f14c84ab
[174] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[175] https://tzworks.com/prototypes/cafae/cafae.users.guide.pdf
[176] https://github.com/EricZimmerman/RECmd/blob/master/BatchExamples/Kroll_Batch.reb

# NetworkList (Wireless Network Profiles List)

"NetworkList" is a registry key[177] that holds information regarding profiles of wireless networks to which the specific machine has connected. Among the information that is stored we can find: the name of the network profile, GUID of the network profile, DNS suffix, the physical address (MAC) of the wireless network device, creation data of the profile, the last connection time and the last update time of the connection[178].

Overall, the full location in the registry of the "NetworkList" artifact is the following: "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList". The "Profiles" subkey contains a list of subkeys, each of them is a GUID of a network profile that contains general information about it (as described above) - as shown in the screenshot below. Due to the fact this information is global to the system it is part of the HKLM hive[179].

Lastly, in order to get the MAC address of the default network\DNS suffix for a specific profile we need to search for the profile's GUID under the"Signatures" subkey (of "NetworkList"). The "Signatures" subkey has its own subkeys "Managed" and "Unmanaged". An example for performing the flow explained is shown in the screenshot below.



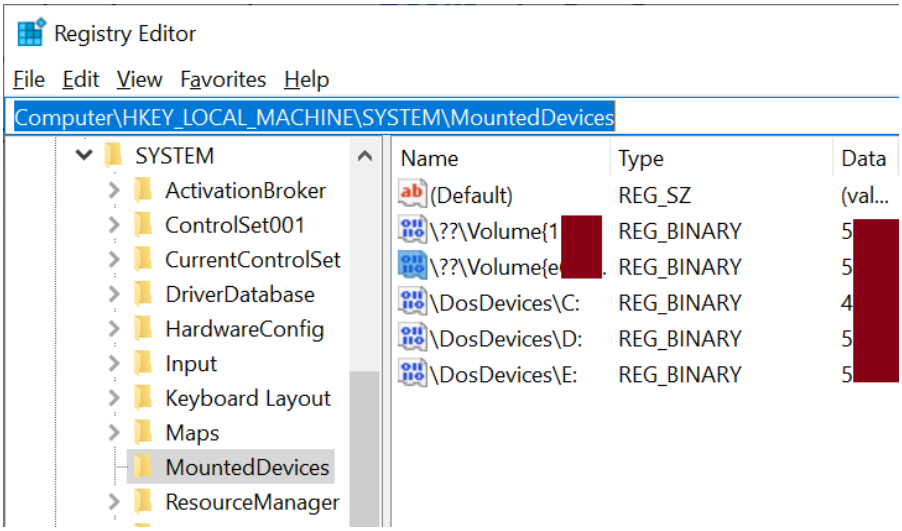

[177] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[178] https://forensafe.com/blogs/winwirelessnetworks.html
[179] https://www.firatboyan.com/en/local-user-profiles-in-windows-10.aspx

# MountedDevices (Drive Letters of Mounted Devices)

"MountedDevices" is a Windows registry key is basically a database which matches serial numbers of USB devices to a given volume/drive letter to when the USB device was connected[180]. The full location of the registry key is: "HKLM\SYSTEM\MountedDevices" - as shown in the screenshot below.

Overall, the "MountedDevices" registry key is used for storing information about devices that have been plugged into a Windows based system. As part of the information which is stored is the drive letter that had been assigned to the drive. This is done to ensure the next the device is plugged it will get the same drive letter assigned[181].

Lastly, we can use the "mountvol.exe" utility in order to remove/create/list a volume mount point[182]. We can think about it as an equivalent to the "mount" utility on Linux[183].



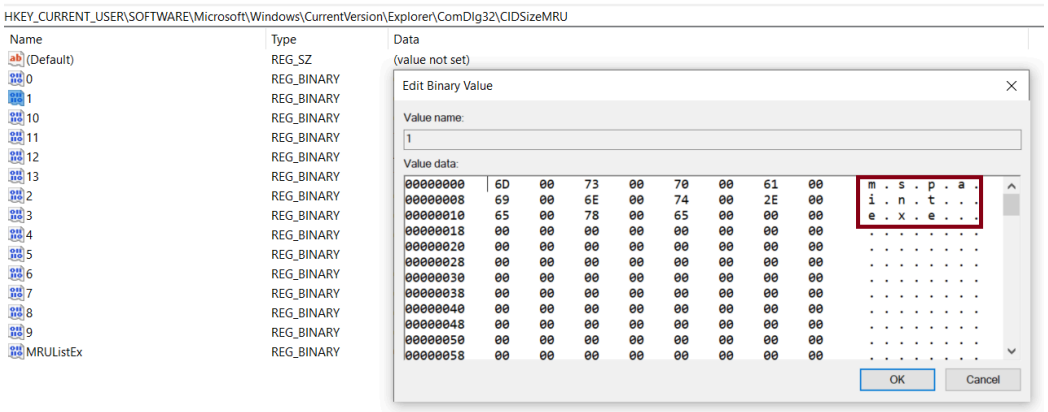

[180] https://www.sciencedirect.com/topics/computer-science/window-registry
[181] https://renenyffenegger.ch/notes/Windows/registry/tree/HKEY_LOCAL_MACHINE/System/MountedDevices/index
[182] https://renenyffenegger.ch/notes/Windows/dirs/Windows/System32/mountvol_exe
[183] https://linux.die.net/man/8/mount

# CIDSizeMRU

"CIDSizeMRU" is a registry subkey of "ComDlg32" which is located in: "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\CIDSizeMRU". This registry key contains a list of recently launched applications[184].

Moreover, it is important to understand that "CIDSizeMRU" does not include all of the launched applications. Only applications that while executing a common dialog box have been opened[185]. The information is recorded when the common dialog box has finished its job.

Lastly, there are different types of common dialog boxes like "Save As", "Open" and "Print"[186]. We can identify applications that can use common dialog boxes by checking if they load "%windir%\System32\comdlg32.dll" (in case of a 64-bit binary) or "%windir%\SysWOW64\comdlg32.dll" (in case of a 32-bit binary).

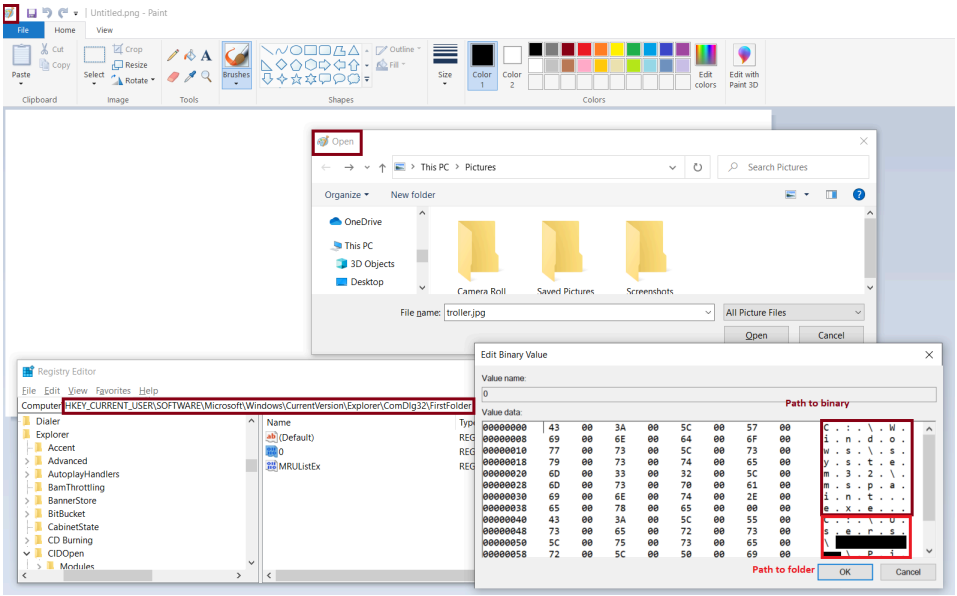



[184] https://docs-cortex.paloaltonetworks.com/r/Cortex-XDR/Cortex-XDR-Pro-Administrator-Guide/Forensics-Add-on-Options
[185] https://learn.microsoft.com/en-us/windows/win32/dlgbox/common-dialog-box-library
[186] https://learn.microsoft.com/en-us/windows/win32/dlgbox/dialog-box-types

# FirstFolder (First Folder Presented During Open/Save As)

The goal of the “FirstFolder” registry key is to track the application's first folder that is presented to the user during an Open or Save As operation. It is a subkey of “ComDlg32” under the registry hive of the user, which is in the following location: “HKCU \SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\FirstFolder”.Also, this information is saved per user as part of its user profile. Thus, we can parse the information from a “NTUSER.DAT” file[187].

Moreover, “FirstFolder” holds the path to the first folder to present while using the common dialog of “Open”/”Save As” and the path of the binary to which it applies - as shown in the screenshot below. Thus, we can identify recently used apps and locations of files used by them[188].



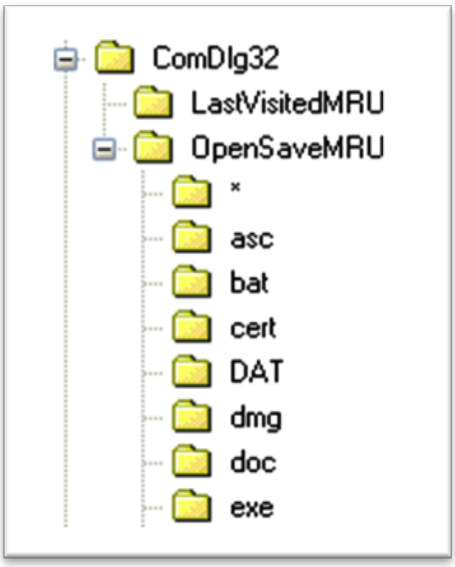

[187] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[188] http://forensicinsight.org/wp-content/uploads/2012/03/INSIGHT-Windows-8-Forensics.pdf

# OpenSaveMRU (Open and Save Most Recently Used)

The goal of the "OpenSaveMRU" registry key is to track files that have been accessed using the "Open" or "Save As" common dialog box in Windows. It is a subkey of "ComDlg32" (HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU) "OpenSaveMRU" is relevant for providing information regarding the usage of different applications like web browsers, graphics editors, audio tools and more[189].

Overall, "OpenSaveMRU" is relevant until Windows XP, since Windows Vista we have "OpenSavePidlMRU"[190]. Moreover, the information is stored in subkeys based on the extensions of the files accessed - as shown in the screenshot below. From "OpenSaveMRU" we can extract the following information: the name of the file accessed using the "Open"\"SaveAs" dialog box, the full path to the file, the order in which the files were accessed, timestamps and more[191].

Lastly, "OpenSaveMRU" information is saved per user as part of its user profile. Thus, we can parse the information from a "NTUSER.DAT" file[192]. By the way, we can also use "OpenSaveMRU" for extracting specific applications/executables used for opening/saving files[193].




[189] https://www.sans.org/blog/opensavemru-and-lastvisitedmru/
[190] https://www.cybertriage.com/artifact/windows-opensave-mru-artifact/
[191] https://forensafe.com/blogs/opensavemru.html
[192] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[193] https://cellebrite.com/en/analyzing-program-execution-windows-artifacts/

# SRUM (System Resource Usage Monitor)

SRUM (System Resource Usage Monitor) is a component of DPS (Diagnostic Policy Service) which helps in troubleshooting, problem detection and resolution for different components in Windows. SURM itself monitors desktop applications/programs/services/windows applications/network connections[194].

Overall, SRUM was first introduced as part of Windows 8 for tracking the system resource usage. The information which is tracked for processes is: CPU cycles, data written/read, network data received/sent, Windows push notification and energy usage. The SRUM data is temporarily stored in the registry, once an hour the data is transferred to an ESE DB located at "%windir%\System32\sru\SRUDB.dat"[195].

Moreover, SRUM has support for extension DLLs which are used for populating different tables in the SRUM database[196]. Among those are: "%SystemRoot%\System32\eeprov.dll" (Energy Estimator Provider), "%SystemRoot%\System32\nduprov.dll" (Network Data Usage Monitor), "%SystemRoot%\System32\wpnsruprov.dll" (Push Notifications (WPN) Provider), "%SystemRoot%\System32\appsruprov.dll" (Application Resource Usage Provider), "%SystemRoot%\System32\ncuprov.dll" (Network Connectivity Usage Monitor) and "%SystemRoot%\System32\energyprov.dll" (Network Connectivity Usage Monitor). There are defined in "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SRUM\Extensions"[197].

Lastly, we can use "SumECmd" created by Eric Zimmerman in order to parse SRUDB.dat and (optionally) SOFTWARE hive for network, process, and energy info[198]. The data can be extracted to CSVs one for each processed table - as shown in screenshot below containing the output of the tool. We can also view the SRUM database using Nirsoft's ESE Database View tool[199]. More about each table and information included in each one is going to be explained on future writeups.


[194] https://lifars.com/wp-content/uploads/2020/09/SRUM-Another-Windows-Time-Machine.pdf
[195] https://0xvighnesh.blogspot.com/2022/01/an-overview-of-srum-forensics.html
[196] https://github.com/WithSecureLabs/chainsaw/wiki/SRUM-Analysis
[197] https://github.com/libyal/esedb-kb/blob/main/documentation/System%20Resource%20Usage%20Monitor%20(SRUM).asciidoc
[198] https://ericzimmerman.github.io/#!index.md
[199] https://www.nirsoft.net/utils/ese_database_view.html

# EventTranscript.db (Windows Diagnostic Database)

In general,"EventTranscript.db" (Windows Diagnostic Database) is an SQLite database which logs lots of diagnostic-related information about events that occur on the Windows operating system in real-time. The file is located at "C:\ProgramData\Microsoft\Diagnosis\EventTranscript\EventTranscript.db"[200]. Telemetry can be defined as the process of sensing and collecting data from a remote system. Microsoft uses telemetry to periodically collect information in order to help improve user experience and for fixing potential issues[201].

Overall, "EventTranscript.db" holds 6 different types of events (called also "tags"). The first, "browsing history", "device connectivity and configuration", "inking typing and speech utterance", "product and service performance" and "product and service usage"[202] - more information about each of those in future writeups. The information collected by "EventTranscript.db" is parallel to the information collected by the eventlog[203]. Thus, clearing the eventlog won't affect the data stored in "EventTranscript.db"

Lastly, we can use tools like "EventTranscriptParser" to extract forensically useful details from "EventTranscript.db". This tool supports extracting the following: Microsoft Edge browsing history, application inventory, Wireless scan results, successful WiFi connection events, User's default preferences (Video player, default browser etc), SRUM information (application execution and network usage) and application execution activity[204] - as shown in the screenshot below.

[200] https://github.com/AndrewRathbun/EventTranscript.db-Research
[201] https://arxiv.org/abs/2002.12506
[202] https://www.kroll.com/en/insights/publications/cyber/forensically-unpacking-eventtranscript
[203] https://medium.com/@boutnaru/the-windows-concept-journey-windows-event-logs-a9945bca421f
[204] https://github.com/stuxnet999/EventTranscriptParser