Microsoft Security

NAVIGATING DFIR STRATEGIES:

Guidance for Incident Responders

Microsoft Incident Response Team's real world insights

Our Guides

How these guides can help in digital forensics and incident response

Guidebook

This set of guides integrate the longstanding expertise of the Microsoft Incident Response team, which has been essential in shaping how we understand Windows internals artifacts in forensic investigations. Since its inception in 2008, this team has been pivotal in defending against a myriad of cyber threats, utilizing the vast Microsoft Threat Intelligence network to aid organizations worldwide .

This expertise is embedded into our resource, designed to highlight key Windows internals artifacts that are crucial yet often misunderstood in forensic contexts. Understanding these artifacts deepens your forensic analysis skills and helps construct detailed activity timelines, enriched by real-world insights from our team at Microsoft.

This set of guides not only boosts analytical capabilities but also contextualizes the evidence within the security frameworks developed by Microsoft over years of incident response .

Microsoft Incident Response supported the actions we'd taken prior to their arrival and helped us expedite the journey ahead.

Director of Information and Communications Technology,
Government of Nunavut

Table of contents (Click titles to navigate)

LEVERAGING WINDOWS INTERNALS FOR FORENSIC INVESTIGATION

Microsoft Security

AmCache Guide

Created by the Microsoft Incident Response Team

The Facts

- A Windows Registry hive that is created to store information related to installed applications, programs executed (or present), drivers loaded, and more
- The primary purpose of AmCache is to enhance the performance and compatibility of applications, ensuring they run smoothly in different environments
- Tracks application files, executed programs, driver binaries, PnP devices, driver packages, device containers, and application shortcuts from Windows 10+ systems
- Executable file name, file path, SHA1 hash, metadata and timestamps are recorded
- The SHA1 hash in AmCache is only calculated for the first 31,457,280 bytes (30 MB) of large files
- Stored in a separate file named Amcache.hve within %SYSTEMROOT%\appcompat\Programs

Common Pitfalls

- AmCache should be considered an "evidence of presence" or "evidence of existence" artifact – it cannot be used to prove a binary executed
- The SHA1 hash in AmCache is NOT always a SHA1 hash of the entire file
    - For large files only the first 31,457,280 bytes (30 MB) are hashed and used as the SHA1 hash within AmCache

Artifact Application

- Use AmCache to show that a file exists (or previously existed) in a given location
- The most interesting AmCache artifacts are those related to files (associated/unassociated file entries)
    - Associated entries are typically part of an installed package whereas unassociated entries are standalone/not part of an installed package
- Use AmCache to acquire the SHA1 hash of a file in a given location but be aware of the limitation on large files
    - Refer to the file size as necessary
- The last write time of the Registry key for a given entry can be used for pivoting and timelining
- Don't immediately discard entries with non-empty product names and/or versions; this metadata can be manipulated as it is taken from the file itself

LEVERAGING WINDOWS INTERNALS
FOR FORENSIC INVESTIGATION

Browser Forensics Guide

Microsoft Security

Created by the Microsoft Incident Response Team

The Facts

- Browser artifacts can be very valuable sources of data for profiling user activity on a system of interest
- History, cookies, cache, session information, settings, and other configuration information may be of forensic interest

- The location of these artifacts varies between Chromium-based browsers such as Microsoft Edge and Google Chrome, and Mozilla Firefox:
    - %LocalAppData%\Microsoft\Edge\UserData\[Default|ProfileX]*
    - %LocalAppData%\Google\Chrome\User Data\Default\*
    - %AppData%\Mozilla\Firefox\Profiles\xxxxxxxx.default-release\*

- History for all three (3) major browsers is stored in a SQLite database within the previously-mentioned locations:
    - "History" for Microsoft Edge and Google Chrome
    - "places.sqlite" for Mozilla Firefox
- Cache for all Chromium-based browsers is stored in a separate directory named "Cache" within the previously-mentioned locations
- Cache for Mozilla Firefox is stored in: %LocalAppData%\Mozilla\Firefox\Profiles\xxxxxxxx.default-release\cache2\*

Common Pitfalls

- Local file access may still be tracked within browser history; this is specifically the case when a browser is used to view a local copy of a PDF, SVG, etc.
- Note that local file access will also appear within %LocalAppData%\Microsoft\Windows\WebCache\WebCacheV01.dat; look for entries like file:///*X:/path/to/file*, where "*X*" is the drive letter on which the file was accessed

Artifact Application

- Never rely on any single tool to parse critical browser history artifacts
- Make sure the tool you are using is compatible with the browser version you are parsing
- Always validate your findings by testing with multiple tools
- Remember that local file access may also appear within browser history records
- The cache may contain copies of downloaded files

Microsoft Incident Response - Your first call, before, during, and after a cybersecurity incident

LEVERAGING WINDOWS INTERNALS FOR FORENSIC INVESTIGATION

Microsoft Security

Link Files & Jump Lists Guide

Created by the Microsoft Incident Response Team

The Facts

- Link files (.lnk) are shortcut files created upon creation of the file with which they are associated; they are primarily used by Windows for the metadata contained within them
- Jump Lists are a collection of Link files
- Upon right-clicking an application within the taskbar, you may see lists of recent files or other information associated with that application; this information is being pulled from the associated Jump Lists
- Link files are located in: %USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent
- Jump Lists are located in:
    - %USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations
    - %USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations

What people get wrong

- Do not ignore Link files and Jump Lists as they hold valuable information that can supplement your forensic investigation
- If a file of the same name is created in a different directory, the target's original path, modified time, and accessed time will be updated in the associated Link file
    - If multiple files of the same name exist, the creation time of the Link file will reflect the creation time of the first instance of that file

How to use the artifact correctly

- Link files can be used as evidence of files that may have once existed but have since been deleted from the device
- Link files contain the Modified, Accessed, and Creation times of the target file
- The creation time of a Link file indicates the first time the target file was created
- The modification time of a Link file indicates the last time the target file was opened
- Other key metadata within a Link file includes target file path, file size, file attributes, origin system name, and origin volume information
- Jump Lists can be useful for seeing files with which a given application has interacted, as well as other relevant information
- Jump Lists contained within AutomaticDestinations pertain to Windows OS provided features common amongst multiple applications; CustomDestinations contain application specific Jump Lists utilized by various applications for a specific purpose
- AutomaticDestinations are in CDF format

Microsoft Incident Response - Your first call, before, during, and after a cybersecurity incident

LEVERAGING WINDOWS INTERNALS FOR FORENSIC INVESTIGATION

Prefetch Guide

Microsoft Security

Created by the Microsoft Incident Response Team

The Facts

- Prefetch is designed to speed up the subsequent launch of applications to improve the overall user experience
- Prefetch is located in %SYSTEMROOT%\Prefetch
- Prefetch is enabled by default on Windows desktop operating systems, but not on Windows Server operating systems
- The prefetching process typically operates within the first ~10 seconds of an application launch, and monitors the files and directories with which an application interacts
- We, as forensic investigators, can leverage Prefetch as a generally reliable evidence of execution artifact
- In Windows 8 and later, the last 8 times of execution are recorded, and the first time of execution can be derived based upon the creation time of the .pf file minus a ~0-10 second delta for the prefetching process
- In Windows 8 and later, 1,024 Prefetch files are kept, using a first in first out process

What people get wrong

- The 8-character hash! The Prefetch hash shown in .pf filenames is computed based upon the file path of the executable and, in some cases, the command line parameters utilized
- For example, notice that you'll see numerous svchost.exe Prefetch files due to the numerous –k flags utilized by svchost.exe
- Prefetch is NOT enabled by default on Windows Server operating systems
- In most cases, you can determine the first time and most recent 8 times of execution for a given binary using Prefetch

How to use the artifact correctly

- The creation time (B) of a .pf file will generally indicate the first time that binary executed on the system (assuming previous Prefetch files were not removed)
- The last modification time (M) of a .pf file will generally indicate the last time that binary executed on the system
- A delta of 0-10 seconds will need to be subtracted from the creation and modification times to account for the Prefetching process time, which varies by application
- Parsing a Prefetch file with a forensic tool can reveal a list of files and directories with which a binary has interacted

LEVERAGING WINDOWS INTERNALS FOR FORENSIC INVESTIGATION

Microsoft Security

ShellBags Guide

Created by the Microsoft Incident Response Team

The Facts

- ShellBags are set of Registry keys which contain details about a user's viewed folder; such as its size, position, and view preferences
- A ShellBags entry is created for every newly explored folder, and it can provide historical evidence of directory access and traversal
- ShellBags are user specific and are found within NTUSER.dat and UsrClass.dat user-specific Registry hives
- ShellBags follow a hierarchy similar to Windows Explorer, with keys providing additional information
- First interacted time, last interacted time, and full folder path (including remote UNC and removable drive paths) are recorded
- .zip files and other compression formats including, but not limited to, .rar, .tar, .tar.gz, etc. are tracked by this artifact because Windows Explorer treats those archives as "folders"; otherwise, files are not tracked within ShellBags

What people get wrong

- With the exception of certain archive formats, files are NOT tracked – only folders
- ShellBags is just for the Windows Explorer GUI! Locations browsed via Command Prompt, PowerShell, or Bash/WSL are NOT tracked by this artifact
- Deleted folders' ShellBags entries are NOT deleted, and they can be updated if new folders share the same name/path
- Exploring a folder is NOT the only way a ShellBags entry is created; changing certain properties on a folder (like the icon) without exploring it will also create an entry
- Desktop.ini is also used to store information about folders; however, the information tracked by Desktop.ini is specific to the folder type (General items, Documents, Pictures, Music, or Videos)

How to use the artifact correctly

- Use ShellBags to show that a folder (or archive) exists (or previously existed) in a given location, that a user had knowledge of a specific folder (or archive), and/or when the folder (or archive) was first or last interacted with and by who
- Existence of ShellBags indicates an interactive session was used, meaning first and/or last interacted timestamps can be used to identify a time period in which an interactive session was active
- Use ShellBags when searching for evidence of data collection and/or staging

LEVERAGING WINDOWS INTERNALS
FOR FORENSIC INVESTIGATION

Microsoft Security

Shimcache Guide

Created by the Microsoft Incident Response Team

The Facts

- Provides compatibility for older software running in newer versions of Windows
- Executive file name, file path, and timestamp are recorded
- Timestamp is the last modification time of the file (M in MACB)
- Windows 7/8/8.1 had a flag that could be used to determine if a program had executed on a system of interest
- You can NOT use Shimcache to prove execution in Windows 10 or later
- Files visible in Windows Explorer windows can determine what is shimmed
- Renaming or moving a file will cause it to be re-shimmed
- Last 1,024 entries are retained
- Most tools will output data with most recently shimmed entries at the top of the list; a numerical cache entry position is also recorded
- Stored in the SYSTEM registry hive
- Only written on reboot or shutdown

Common Pitfalls

- The timestamp! It is NOT the time of shim and it is NOT the time of execution.
- For Windows 10 and later, this artifact should be considered an “evidence of presence” or “evidence of existence” artifact – it can not be used to prove a binary executed.
- The data within Shimcache is only written to disk upon reboot or shutdown. Unless you capture memory and attempt to extract Shimcache via a Volatility plugin (or other means), that data will not be available until the next reboot.

Artifact Application

- Use Shimcache to show that a file exists (or previously existed) in a given location. While it was previously considered to be an “evidence of execution” artifact prior to Windows 10, it would be best described as an “evidence of presence” or “evidence of existence” artifact.
- The cache entry position can be used to show potential items of interest. The lower the cache entry position number, the more recently that binary was shimmed.
- Remember that if a file is renamed or moved, it will be re-shimmed. If file a.exe was at cache entry position 10, and file scvhost.exe was at cache entry position 5, and both files had the same 64-bit modification (M) timestamp, you could say with high confidence that binary a.exe was renamed to scvhost.exe.
- Context matters! Look for items of interest in close proximity to one another. If you find a known malicious binary in the Shimcache, the cache entry positions just before and after that binary may reveal additional items of interest.

LEVERAGING WINDOWS INTERNALS FOR FORENSIC INVESTIGATION

Microsoft Security

SRUM Guide

Created by the Microsoft Incident Response Team

The Facts

- System Resource Utilization Monitor (SRUM) was introduced in Windows 8 and Windows Server 2019 and was designed to track the utilization of various system resources such as CPU usage, network activity, and even battery consumption
- Some of the details collected as part of SRUM can be viewed in the App History tab within Task Manager, but there is much more information not displayed in the GUI
- SRUM tracks key information regarding application execution such as the name and path of every executed application on the system and the SID of the identity that executed the application, even if the application has since been deleted
- Other useful data stored in SRUM includes the names of networks that have been connected to, CPU usage by foreground and background operations, and bytes read and written from the hard drive by applications
- SRUM data is stored in a Windows ESE database located in the following file: %SYSTEMROOT%\System32\sru\SRUDB.dat
- SRUM is only available on Windows 8 and later and Windows Server 2019 and later

What people get wrong

- Since SRUM, like most artifacts, wasn't introduced for a forensic-specific purpose, it is often overlooked in forensic documentation and by investigators alike
- SRUM can be used as one of the few reputable artifacts of execution and should be a vital part of any system analysis
- The timestamps within SRUM can NOT be used to determine the time an application was executed on a system; however, you can determine a general timeframe
- On Windows Server, the network data usage of SRUM is NOT populated like it is on Windows Desktop counterparts

How to use the artifact correctly

- SRUM tracks system application usage and user identities through SIDs, linking executed apps to user activities. This data can reveal unauthorized or harmful software executions tied to specific accounts, aiding in investigations.
- The network-related information stored in SRUM can tell an investigator where a particular device connected; SSID names and specific byte transfers inbound and outbound along with their associated applications may be available
- Use SRUM Network Data Usage to show data transfer by a given process, but keep in mind that this information is only available on Desktop operating systems

LEVERAGING WINDOWS INTERNALS FOR FORENSIC INVESTIGATION

User Access Logging Guide

Microsoft Security

Created by the Microsoft Incident Response Team

The Facts

- User Access Logging (UAL) is enabled by default on Windows Server operating systems, starting with 2012 and later
- Collects user access and system-related statistical data in near real-time
- Examples of services and roles from which data is collected include DNS, DHCP, IIS, WSUS, etc.
- Stored within multiple .mdb files (ESE databases) located in %SYSTEMROOT%\System32\LogFiles\SUM
- SystemIdentity.mdb, Current.mdb, and one or more files with a GUID-based name should exist in this location
- The GUID-based file names will hold data from the current year, the previous year, and two (2) years prior
- Every 24 hours, data from Current.mdb will be copied to the GUID-named database for the current year
- SystemIdentity.mdb will track the other UAL databases and contain basic server configuration info

What people get wrong

- This artifact is only present on Windows Server operating systems, 2012 and later
- The IP address tracked is the location from which the associated activity originated; the destination is the Windows Server system from which UAL was obtained
- On the first day of the year, UAL will create a new GUID-named .mdb file
    - The old GUID-named file is retained as an archive; after two (2) years, the original GUID.mdb will be overwritten
- The activity is tracked by server role, which maps to the roles configured on the Windows Server from which the data was acquired

How to use the artifact correctly

- This artifact can be used to identify abnormal access to systems and to profile lateral movement from various clients to servers running Windows Server 2012 or later
- The InsertDate is logged in UTC, and represents the first access for the year for a combination of the specific user, source IP address, and role
- The LastAccess is logged in UTC, and represents the last access for the year for a combination of the specific user, source IP address, and role
- The "File Server" role is usually associated with SMB access, but in some cases, access via other protocols may be associated with this role

LEVERAGING WINDOWS INTERNALS FOR FORENSIC INVESTIGATION

Microsoft Security

UserAssist Guide

Created by the Microsoft Incident Response Team

The Facts

- UserAssist records metadata on GUI-based program executions
- Metadata is stored in the NTUSER.dat user-specific Registry hive and is ROT-13 encoded
- GUIDs identify the type of execution (Win7+)
    - CEBFF5CD - Executable File Execution
    - F4E57C4B - Shortcut File Execution
- Application path, last executed timestamp, run count, focus time, and focus count are recorded
- The number of entries that can be saved in the UserAssist Registry key is limited by the size of the Registry
- UserAssist may be able to show execution on Windows 10 and later via the Focus Time

What people get wrong

- For Windows 10 and later, this artifact can still be used to prove execution, but attention needs to be on the Focus Time, NOT the Run Count
- For Windows 10 and later, right clicking an application on the Start Menu and selecting "Open file location" will cause an entry to be created/updated in UserAssist for the Executable file execution type. This action does not cause a binary to execute but does cause the Run Count to increase and the Last Executed timestamp to update. However, the Focus Time will NOT be updated.
- The Run Count and Last Execution Time can be manipulated as mentioned above, so do NOT use it to prove how many times a program executed or when it last executed.

How to use the artifact correctly

- Use UserAssist to prove that a binary was executed by looking at the Focus Time being greater than zero (0). Keep in mind that certain binaries are executed without user interaction.
- When UserAssist entries are seen for Snipping Tool.lnk and Paint.lnk for a single user with the same Last Executed timestamp, this is indicative that an interactive session likely occurred for that user for the first time ~100 seconds after the specified timestamp. This does not apply for Windows 11.
- Understand how entries are created in UserAssist for binaries of interest depending on how they are executed. For example, UserAssist entries will not be created for certain binaries if execution was done through a web browser's downloads dialog.

Microsoft Security

Thank you.

Links and other resources:

→ Navigating the maze of IR

→ Meet the Microsoft Incident Response team

→ Learn more about Microsoft Incident Response

→ Microsoft Security Resources