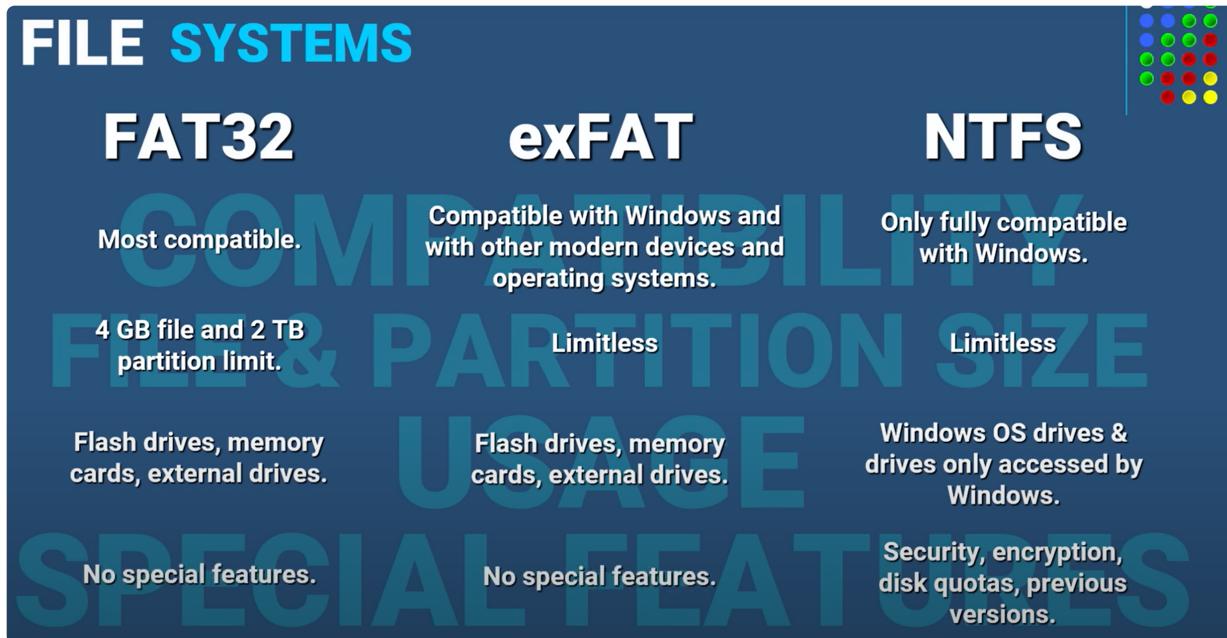


Labs - Introduction to Digital Forensics

Preliminary knowledge

- Forensics image: An direct copy of a physical storage device (including computers).
- File System: Organised storage of data on computer for efficiency, compatibility...
- FAT: Known as File allocation table, it's the earliest file system and the most compatible File System but support file only up to 4gb and supports max partition of 2TB
- exFat: Known as extended File allocation table, it's an improvement over FAT and supports max partition size of much much over 2TB (limitless) as well as file size but not as compatible as FAT.
- NTFS: Known as New Technology File system, it's benefit over FAT is unlimit File and disk partition size and become particular after release of Windows XP. It also supports many features than FAT32 and exFAT (such as security and previous versions (shadow copy) on the file tab as an illustration) but has the least compatibility of the three.



<https://www.youtube.com/watch?v=bYjQakUxeVY>

- MFT: Known as master file table, it's a file that contains the file metadata for the NTFS file system.
 - Cool thing is that when files are deleted (e.g. from an NTFS system volume), the MFT entries are marked as free and may be reused (so recovery isn't always possible)
<https://www.youtube.com/watch?v=h8Mb55ox5OE>
- MFT entry sequence number (seems to be an sequence number that conveniently maps to the MFT file), also known as the ID of this record:

MFT Explorer v2.0.0.0

File Tools Help

Name	Image Icon	Name	Parent Path	Is Dir	Is Deleted	SI_Created On	FN_Created On	SI_Modified On	FN_Modified On
C:\Users\johndoe\Desktop\forensic	No image data	ChangedFileTime.txt	\Temp			2022-01-03 16:54:25.2726453	2023-09-07 08:30:12.4258743	2023-09-07 08:30:12.4258743	
		creds.txt	\Temp			2023-09-07 08:31:15.6284444		2023-09-07 08:31:26.142547	2023
		discord.exe	\Temp		<input checked="" type="checkbox"/>	2023-09-07 08:28:48.7170313		2023-09-07 08:28:51.0725203	
		groups.txt	\Temp			2023-09-07 08:30:26.9097759		2023-09-07 08:30:26.9567342	2023
		ipinfo.txt	\Temp			2023-09-07 08:30:26.9724020		2023-09-07 08:30:27.0033627	2023
		networkinfo.txt	\Temp			2023-09-07 08:30:27.0033627		2023-09-07 08:30:27.0502741	2023
		pass.exe	\Temp			2023-09-07 08:28:52.8586497		2023-09-07 08:28:57.4169173	
		pass.ps1	\Temp			2023-09-07 08:28:59.0670101		2023-09-07 08:29:04.6964247	

Properties

- Copied
- Has ADS
- Is deleted
- Is directory
- Possible Timestamped

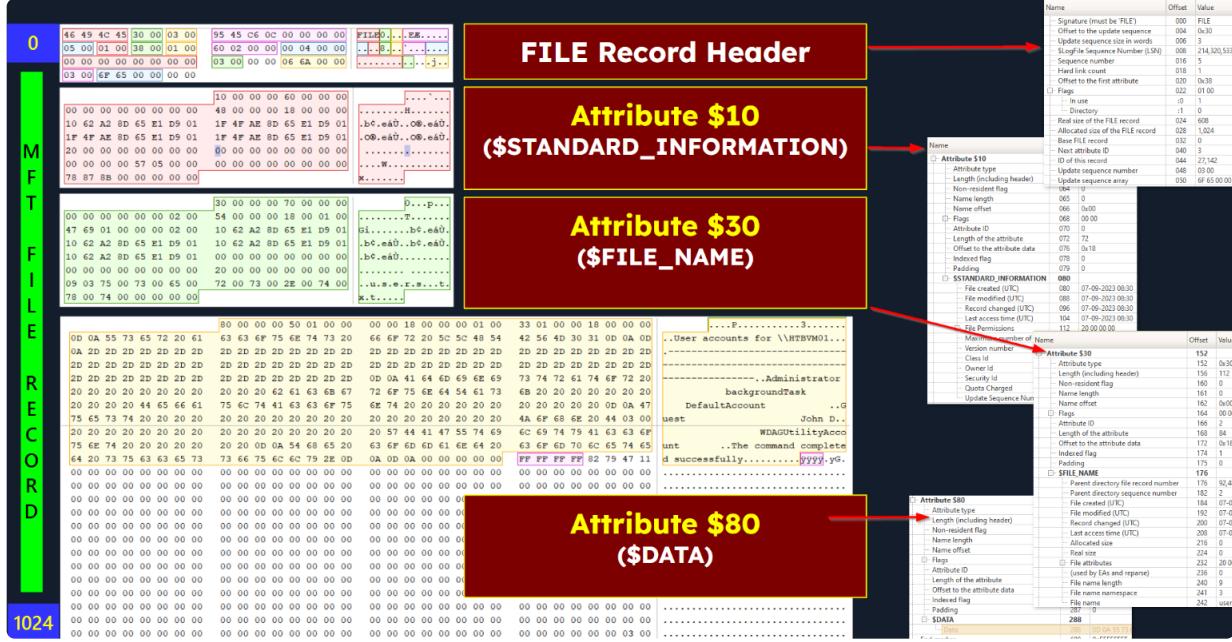
Selected directory: \Temp

```

Usage:
MFTECmd [options]

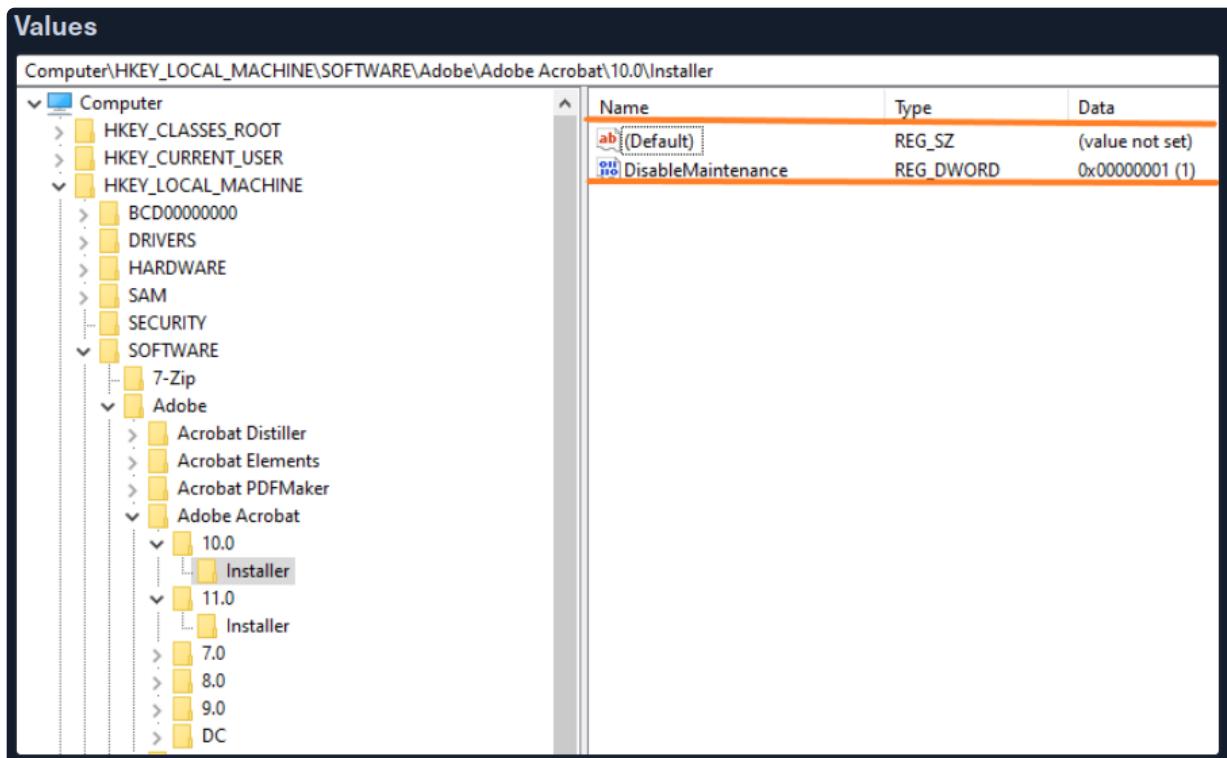
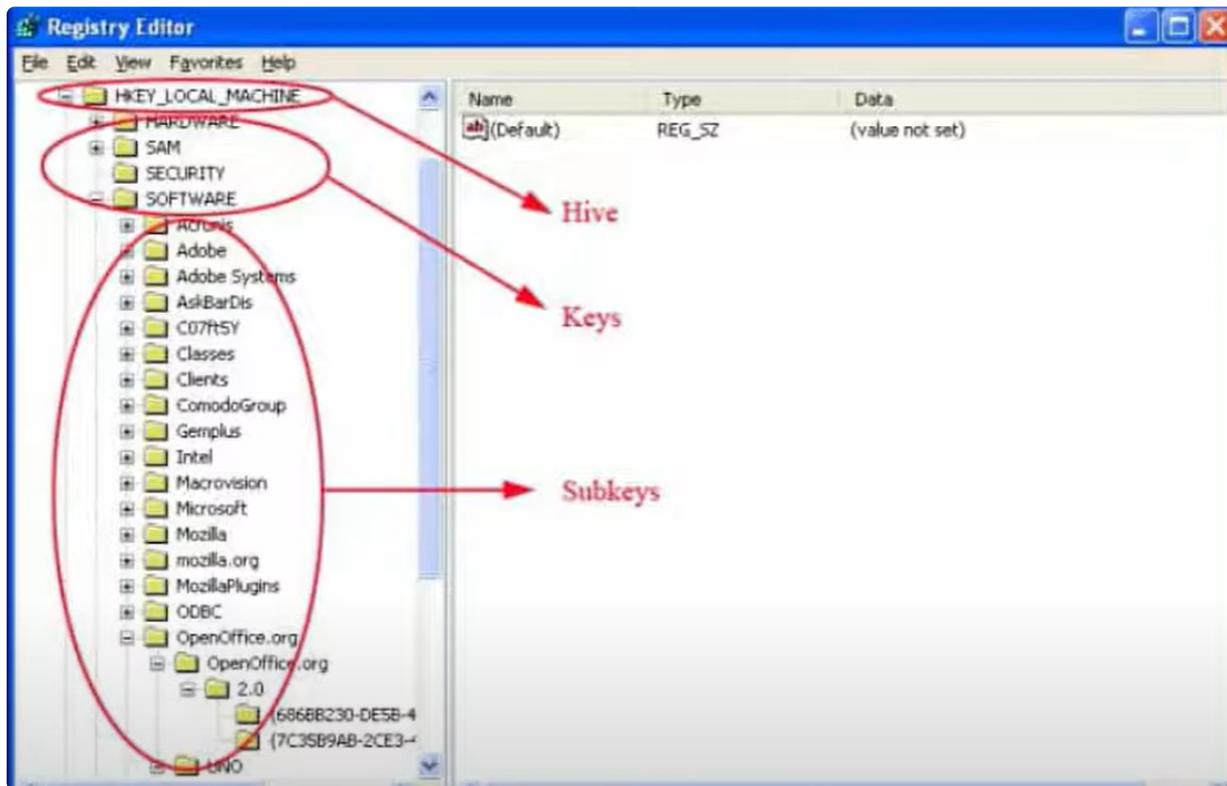
Options:
-f <f>           File to process ($MFT | $J | $Boot | $SDS | $I30). Required
-m <m>           $MFT file to use when -f points to a $J file (Use this to resolve parent path in $J CSV output)
--json <json>     Directory to save JSON formatted results to. This or --csv required unless --de or --body is specified
--jsonf <jsonf>   File name to save JSON formatted results to. When present, overrides default name
--csv <csv>       Directory to save CSV formatted results to. This or --json required unless --de or --body is specified
--csvf <csvf>    File name to save CSV formatted results to. When present, overrides default name
--body <body>     Directory to save bodyfile formatted results to. --bdl is also required when using this option
--bodyf <bodyf>   File name to save body file formatted results to. When present, overrides default name
--bdl <bdl>       Drive letter (C, D, etc.) to use with bodyfile. Only the drive letter itself should be provided
--blf             When true, use LF vs CRLF for newlines [default: False]
--dd <dd>         Directory to save exported $MFT FILE record. --do is also required when using this option
--do <do>         Offset of the $MFT FILE record to dump as decimal or hex. Ex: 5120 or 0x1400 Use --de or --debug to see offsets
--de <de>         Dump full details for $MFT entry/sequence #. Format is 'Entry' or 'Entry-Seq' as decimal or hex.
Example: 5, 624-5 or 0x270-0x5.
--dr              When true, dump $MFT resident files to dir specified by --csv, in 'Resident' subdirectory. Files will be named '<EntryNumber>-<SequenceNumber>_<FileName>.bin'
--fls             When true, displays contents of directory from $MFT specified by --de. Ignored when --de points to a file [default: False]
--ds <ds>         Dump full details for Security Id from $SDS as decimal or hex. Example: 624 or 0x270
--dt <dt>         The custom date/time format to use when displaying time stamps. See https://goo.gl/CNVq0k for options [default: yyyy-MM-dd HH:mm:ss.fffffffff]
--sn              Include DOS file name types in $MFT output [default: False]
--fl              Generate condensed file listing of parsed $MFT contents. Requires --csv [default: False]
--at              When true, include all timestamps from 0x30 attribute vs only when they differ from 0x10 in the $MFT [default: False]
--rs              When true, recover slack space from FILE records when processing $MFT files. This option has no effect for $I30 files [default: False]
--vss             Process all Volume Shadow Copies that exist on drive specified by -f [default: False]
--dedupe          Deduplicate -f & VSCs based on SHA-1. First file found wins [default: False]
--debug           Show debug information during processing [default: False]
--trace           Show trace information during processing [default: False]
--version         Show version information
-?, -h, --help    Show help and usage information

```



- File slack: Unused portion of storage given its memory allocation. The pudding and bowel illustration is an great explanation of this.
<https://www.canto.com/blog/file-slack/>
<https://www.techrepublic.com/article/anatomy-of-hard-disk-clusters/>
- File signature: An Specific byte sequence unique to a file
<https://nordvpn.com/cybersecurity/glossary/file-signature/>
- USN Journal: Known as the update sequence number journal maintains change logs (creation, deletion and modification details) to file made to NTFS or ReFs volumes.
- Volumes: A volume is also known as a logical drive, where it is an accessible storage area with a single file system.
- A partition is a logcial separation of a hard disk.
<https://forensafe.com/blogs/usnjournal.html>
<https://learn.microsoft.com/en-us/windows-hardware/drivers/ifs/storage-device-stacks--storage-volumes--and-file-system-stacks>
[https://en.wikipedia.org/wiki/Volume_\(computing\)](https://en.wikipedia.org/wiki/Volume_(computing))
- .LNK Files: It is known as a Windows short cut, which points to and is used to open another file, folder or application.
<https://fileinfo.com/extension/lnk>
- Prefetch file: Files created by Windows OS to speed up loading time of applications through caching the necessary data for frequently used program.
- Cache: A smaller amount of faster but expensive memory for accessing data.
<https://www.geeksforgeeks.org/prefetch-files-in-windows/>
<https://www.techtarget.com/searchstorage/definition/cache>
- Windows registry: A hierarchical database that stores low-level settings for Microsoft Windows OS and for applications that uses the registry.

- Registry Hives: A logical group of keys, subkeys and values in the registry that has a set of supporting file loaded into memory when the OS is started or a user logs in.



Registry Hives

Each Windows host has a set of predefined Registry keys that maintain the host and settings required for use. Below is a breakdown of each hive and what can be found referenced within.

Hive Breakdown

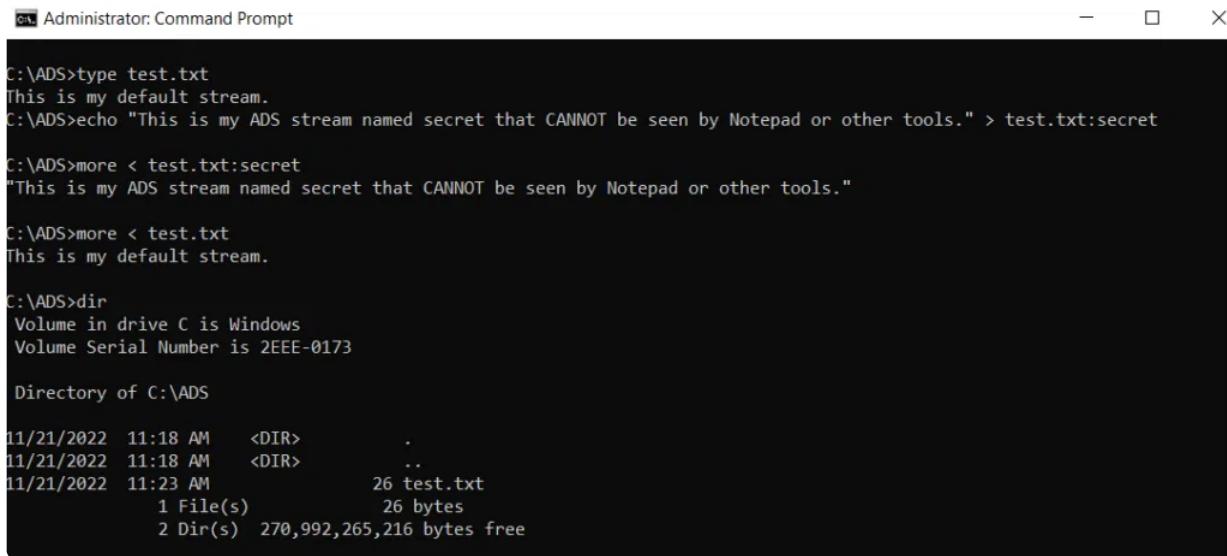
Name	Abbreviation	Description
HKEY_LOCAL_MACHINE	HKLM	This subtree contains information about the computer's physical state , such as hardware and operating system data, bus types, memory, device drivers, and more.
HKEY_CURRENT_CONFIG	HKCC	This section contains records for the host's current hardware profile . (shows the variance between current and default setups) Think of this as a redirection of the HKLM CurrentControlSet profile key.
HKEY_CLASSES_ROOT	HKCR	Filetype information, UI extensions, and backward compatibility settings are defined here.
HKEY_CURRENT_USER	HKCU	Value entries here define the specific OS and software settings for each specific user. Roaming profile settings, including user preferences, are stored under HKCU.
HKEY_USERS	HKU	The default User profile and current user configuration settings for the local computer are defined under HKU.

<https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry-hives>

https://en.wikipedia.org/wiki/Windows_Registry

- Shellbags: Shellbags are set of registry keys which contain details about a user's viewed folder, such as its size, position and icon.
<https://medium.com/ce-digital-forensics/shellbag-analysis-18c9b2e87ac7>
<https://www.youtube.com/watch?v=YvVemshnpKQ>
- Thumbnail cache: It stores thumbnail images for for Window's Explorer's thumbnail view, so these images don't need to be calculated everytime it is viewed.
- File Explorer/Window's explorer: A file manager application that provides an GUI access for accessing file systems.
https://en.wikipedia.org/wiki/Windows_thumbnail_cache
https://en.wikipedia.org/wiki/File_Explorer
- Alternative Data Streams (ADS): An file stream that does not belong to the main data which can allow malicious data about a file hidden from the user. It is usually created manually.
 - Generally it is mostly used for zone.identifier for identifying web downloads in digital forensics.

E.g. Alternative data stream of secret.



```
C:\ADS>type test.txt
This is my default stream.
C:\ADS>echo "This is my ADS stream named secret that CANNOT be seen by Notepad or other tools." > test.txt:secret
C:\ADS>more < test.txt:secret
"This is my ADS stream named secret that CANNOT be seen by Notepad or other tools."
C:\ADS>more < test.txt
This is my default stream.

C:\ADS>dir
Volume in drive C is Windows
Volume Serial Number is 2EEE-0173

Directory of C:\ADS

11/21/2022  11:18 AM    <DIR>      .
11/21/2022  11:18 AM    <DIR>      ..
11/21/2022  11:23 AM           26 test.txt
                           1 File(s)       26 bytes
                           2 Dir(s)  270,992,265,216 bytes free
```

- File stream: An sequence of bytes about an data. By default there is only one main stream (\$DATA or "").

<https://www.malwarebytes.com/blog/news/2015/07/introduction-to-alternate-data-streams>

https://blog.netwrix.com/2022/12/16/alternate_data_stream/

- Security identifier (SID): A unique value of variable length to identify a trustee (user account, group account, logon session) to which an access control entry (ACE).
- Security descriptor: A security descriptor contains security information (e.g. SIDS, DACL, SACL...) associated with a securable object
- Securable Object: An object that can have a security descriptor, where all named Windows object are securable.
 - Some e.g. of named object (?):

Object type	Security descriptor functions
Files or directories on an NTFS file system	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Named pipes Anonymous pipes	GetSecurityInfo , SetSecurityInfo
Processes Threads	GetSecurityInfo , SetSecurityInfo
File-mapping objects	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Access tokens	SetKernelObjectSecurity , GetKernelObjectSecurity
Window-management objects (window stations and desktops)	GetSecurityInfo , SetSecurityInfo
Registry keys	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Windows services	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Local or remote printers	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Network shares	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Interprocess synchronization objects (events, mutexes, semaphores, and waitable timers)	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Job objects	GetNamedSecurityInfo , SetNamedSecurityInfo , GetSecurityInfo , SetSecurityInfo
Directory service objects	These objects are handled by Active Directory Objects. For more information, see Active Directory Service Interfaces .

<https://learn.microsoft.com/en-us/windows/win32/secauthz/security-descriptors>

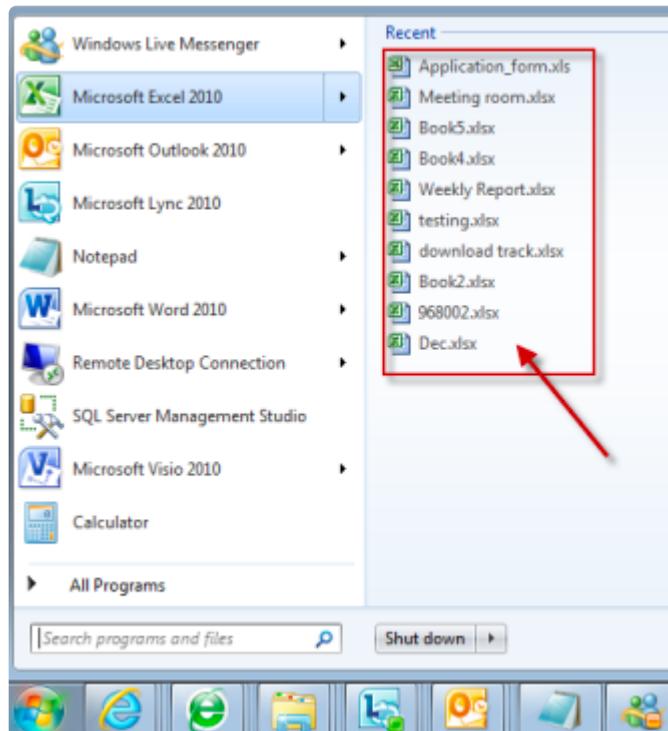
<https://learn.microsoft.com/en-us/windows/win32/secauthz/trustees>

<https://learn.microsoft.com/en-us/windows/win32/secauthz/securable-objects>

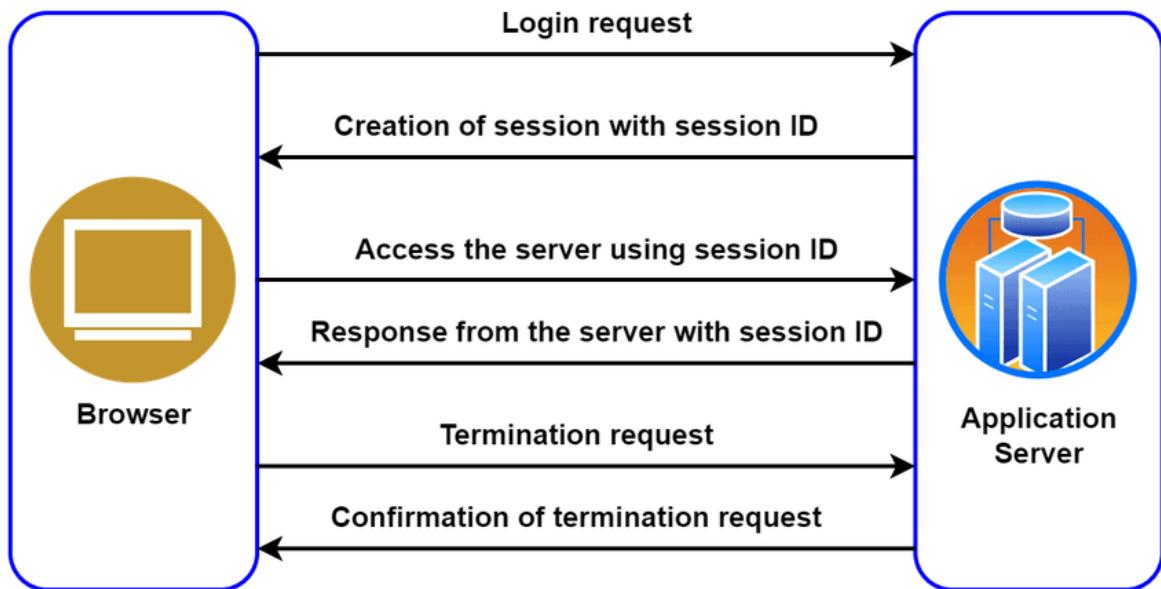
- Shimcache/AppCompatCache: It stores details about program execution details, including full file path, file size, process execution flag and is stored in the registry.
 - It is designed for installed programs/executables (does not necessarily need to be executed) and uses this info for troubleshooting with Compatibility mode.
 - Amcache: It stores information about program executions, including full execution path, deleted time, file hashes... and is stored in a registry or BCF file.
 - It is designed for applications that have been executed.
- <https://medium.com/@mehrnoosh/shimcache-amcache-forensic-analysis-99a8a9733772>
- https://www.reddit.com/r/digitalforensics/comments/uvosan/the_difference_between_shimcache_amchachehve/

- UserAssist: A feature in windows that tracks the usage of executable files and application launched by the user.
<https://www.magnetforensics.com/blog/artifact-profile-userassist/>
- RunMRU list: Known as the Run Most Recently used list is an window registry that stores most recently run/executed program.
https://docs.velociraptor.app/artifact_references/pages/windows.timeline.registry.runmru/
- Jump List: Jump list is a feature that allows user to view most recently accessed or used files for specific applications.

E.g. Jump list for Microsoft Excel 2010



- Registry run keys: Adding references to program of a run key causes the program referenced to run at startup and is often used by malicious actors.
<https://attack.mitre.org/techniques/T1547/001/>
- Session: An mechanism that uses an unique ID to identify the user after it logs in (for web)



In a world without cookies/ sessions....

I'm calling to confirm whether
you changed my address like
I asked you last Monday?



Ummm....excuse
me sir, I have no
idea who you are?



In a world with cookies/session...

Changed my address?
Here's my ID card.



Of course, let me
bring up your
records to find out
who you are.



I see that you are
Brian May. Yes, we
can confirm a
change of
address. Thank you
for providing us with
your ID



your ID



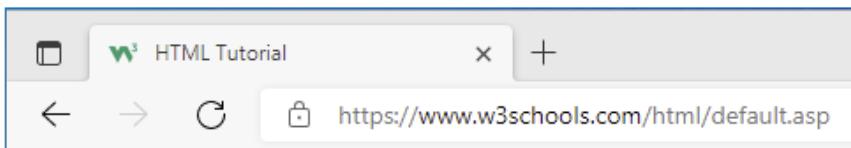
(By Ben Koshy, BKSpurgeon) - and yes, Queen
is my fav band in the whole world!!

<https://stackoverflow.com/questions/3804209/what-are-sessions-how-do-they-work>

<https://www.baeldung.com/cs/web-sessions>

- Favicon: A favicon is a small image on the left next to the page title on the left.

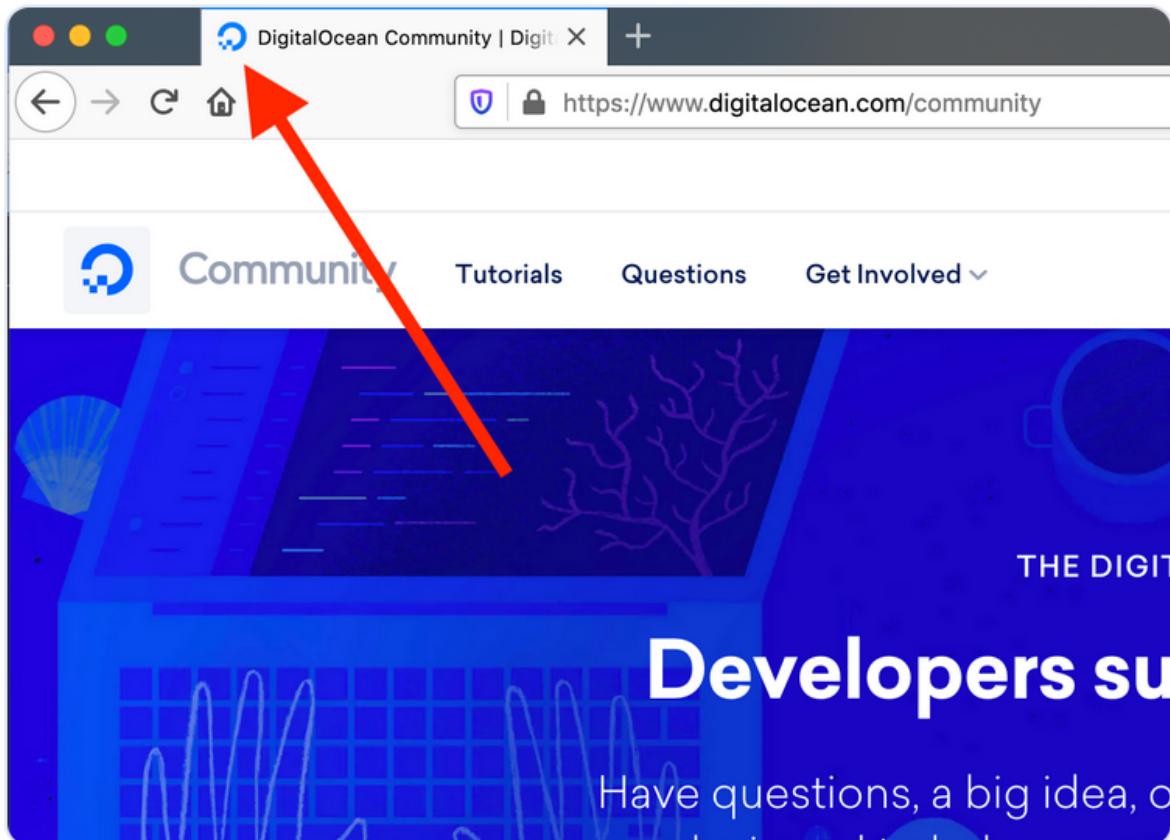
A favicon image is displayed to the left of the page title in the browser tab, like this:



To add a favicon to your website, either save your favicon image to the root directory of your website, and save your favicon image in this folder. A common name for a favicon image is "favicon.ico".

Next, add a `<link>` element to your "index.html" file, after the `<title>` element, like this:

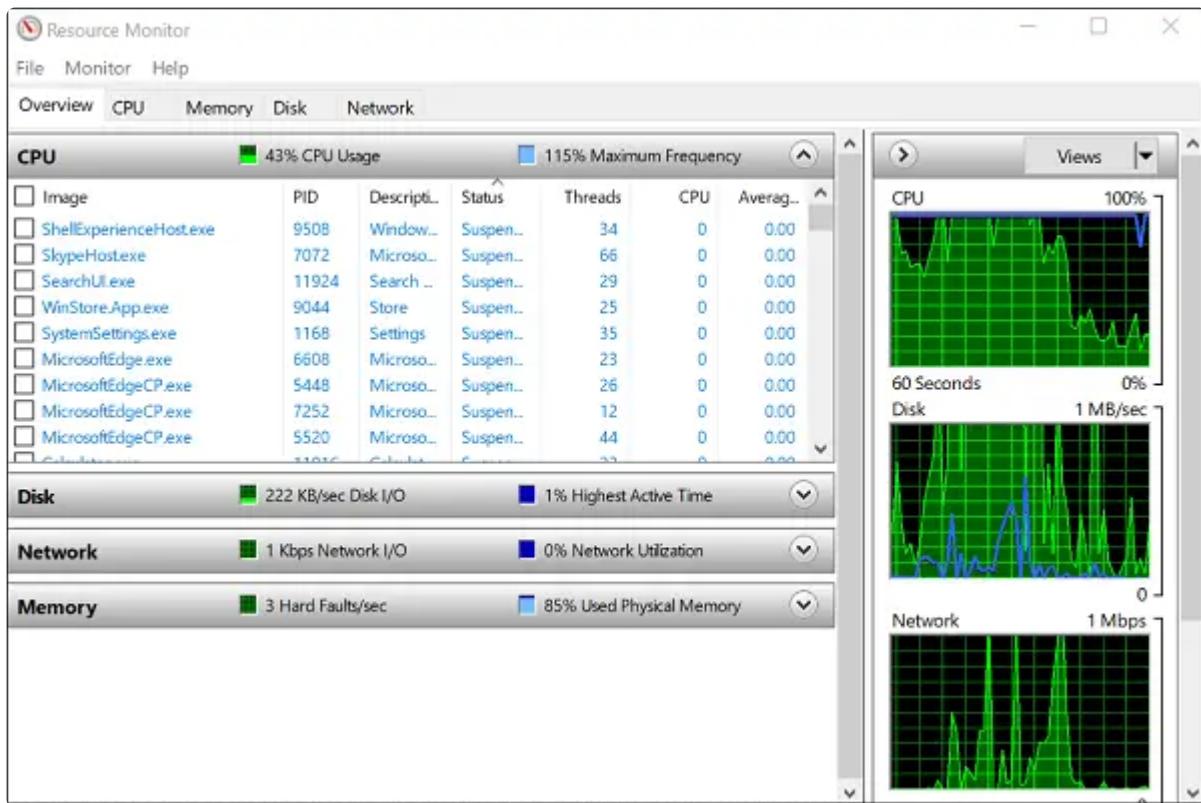
A favicon is a small image that is located in the browser tab to the left of a webpage's title. The image below illustrates the location of a favicon.



<https://www.digitalocean.com/community/tutorials/how-to-add-a-favicon-to-your-website-with-html>

https://www.w3schools.com/html/html_favicon.asp

- Tab recovery data: A feature that modern browsers implement and supports recovery of browser crash.
<https://helpdeskgeek.com/how-to/how-to-recover-closed-tabs-in-any-web-browser/>
- SRUM: Known as System resource usage monitor, it tracks resource utilisation and application usage pattern.
E.g. It looks like the following:



<https://www.thewindowsclub.com/use-resource-monitor-windows-10>

- SRUDB: It is the SRUM database and is also the same kinda of database used in AD, it is being written hourly and at system shutdown.

<https://www.magnetforensics.com/blog/srum-forensic-analysis-of-windows-system-resource-utilization-monitor/>

- KAPE: A useful solution for extracting useful artifacts for forensic analysis.
- Artifacts: Pieces of evidence that maybe useful for our analysis.
- Handle: A handle in Windows is an abstract reference value to the resources, often memory or an open file, or a pipe.

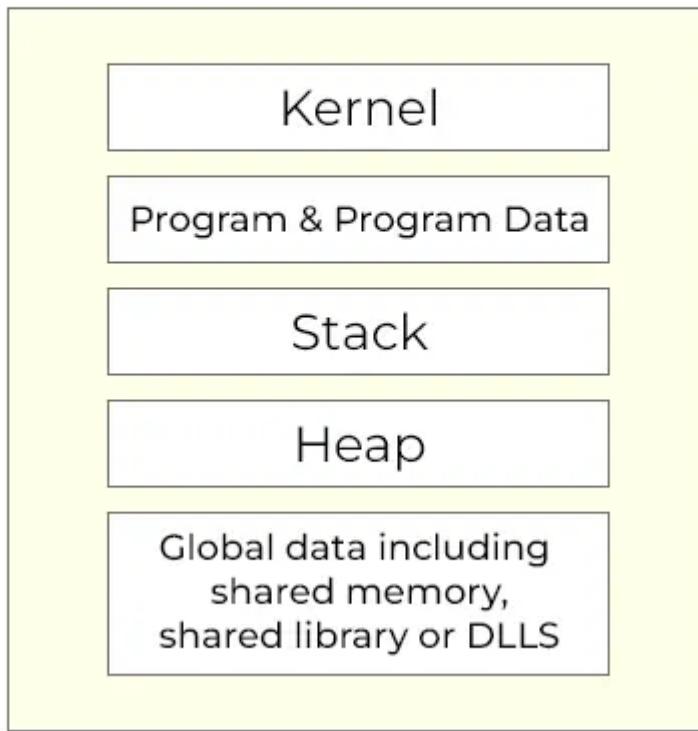
<https://stackoverflow.com/questions/902967/what-is-a-windows-handle>

- Beaconing: Beaconing is a term where malware periodically sends brief message from an infected host to an attacker's machine and is waiting for further instructions.

<https://healthybyte.net/cybersecurity/what-is-malware-beaconing>

- Process hollowing: An technique where adversaries inject malicious code into suspended and hollowed process. It is a method of executing arbitrary code in the address space of a separate live process
- Process address space: The amount and space of memory allocated to a process.
E.g. For an OS

Process Address Space in OS



<https://prepinsta.com/operating-systems/process-address-space/>

<https://attack.mitre.org/techniques/T1055/012/>

- Rootkit: A rootkit malware is an collection of software designed to give malicious actors control of a computer network or applications. They may also set up further exploits and deliver additional malware.
<https://www.crowdstrike.com/cybersecurity-101/malware/rootkits/>
- Volatility: Open source tool for conducting memory forensic. Useful documentations are as follows:
<https://github.com/volatilityfoundation/volatility/wiki/Command-Reference>
<https://github.com/volatilityfoundation/volatility/wiki/Command-Reference-Mal>
<https://blog.onfvp.com/post/volatility-cheatsheet/>
- Doubly linked list: A doubly linked list is a datastrcutre where it consist a set of nodes, each of which contains a value and two pointers.

Doubly Linked List in Data Structure



26

<https://www.geeksforgeeks.org/doubly-linked-list/>

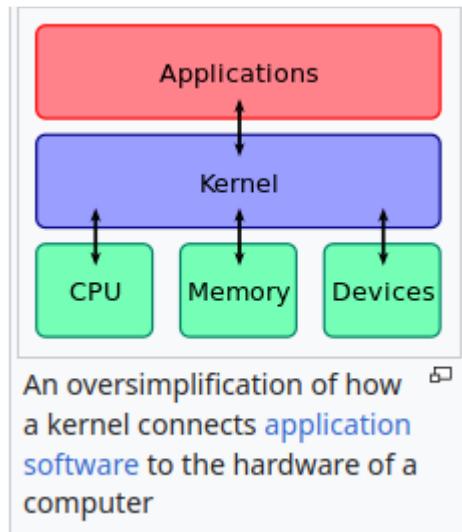
- WoW64: Known as Windows 32-bit on Windows 64-bit, is a subsystem of Windows OS that can run 32-bit applications on 64-bit Windows.
 - It is included in all 64-bit versions of Windows.
- 32bit vs 64bit: it is where data units (such as integers, memory addresses) are in 32 or 64 bit units respectively for the processing unit of the computer.
<https://en.wikipedia.org/wiki/WoW64>
<https://www.lifewire.com/32-bit-64-bit-2624554>
- Win7SP1x64: An specific windows version known as Windows 7 Service pack 1 (SP1)
- Memory pool: It is a logical division of main memory that is reserved for processing a job or group of jobs.
- Pool tag: A 4 byte character that is associated with a dynamically allocated chunk of pool memory. It is specified by a driver when it allocated the memory.
 - Pool tag scanning can help find process that previously terminated or have been hidden or unlinked by a rootkit.
 - But this pool tag can also be overwritten, but not often seen.

<https://www.geeksforgeeks.org/what-is-a-memory-pool/>

<https://techcommunity.microsoft.com/t5/ask-the-performance-team/an-introduction-to-pool-tags/ba-p/372983>

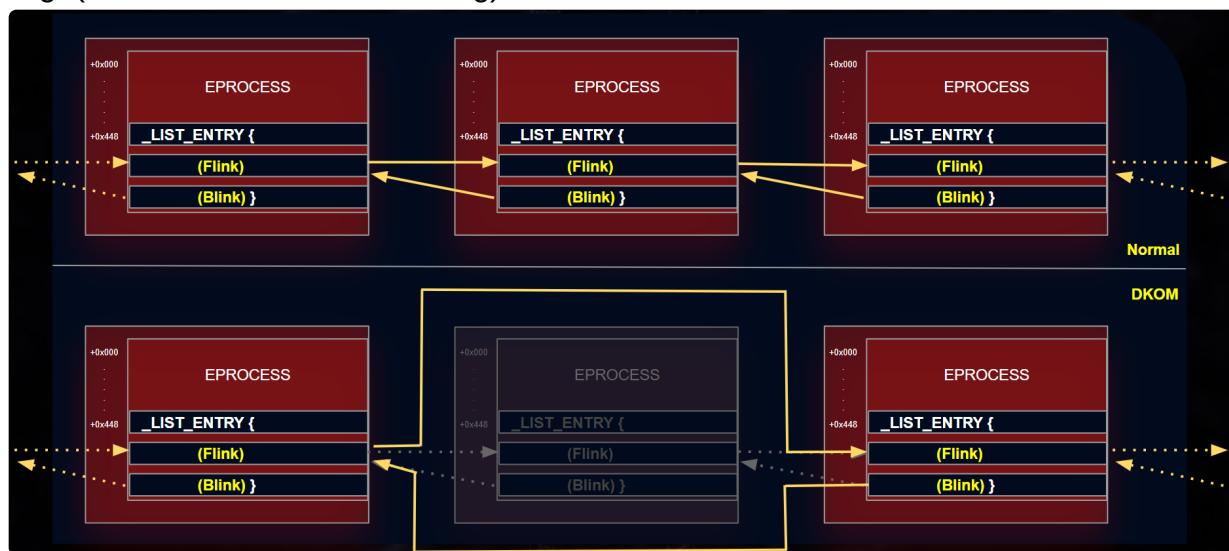
- svchost: A system process that can host one or more windows service on an Windows OS.
<https://en.wikipedia.org/wiki/Svchost.exe>
- LoadCount: Load count is a 16-bit value which tells you whether an DLL is statically linked (if it is treated as an -1 0xffff, otherwise it is dynamically linked).
<https://stackoverflow.com/questions/22855932/is-0x0000ffff-the-default-load-count-of-a-dll-in-windows>

- Static vs dynamic linking: Dynamic link programs at run time, while static copies code the program would use from other files (e.g. object file) at compile time.
<https://earthly.dev/blog/static-and-dynamic-linking/>
- Kernel: The core of an OS and controls over everything in the system



[https://en.wikipedia.org/wiki/Kernel_\(operating_system\)](https://en.wikipedia.org/wiki/Kernel_(operating_system))

- EProcess: A structure that is the Kernel's representation of a process object.
<https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/ntos/ps/eprocess/index.htm>
- Task scheduler: A job scheduler that launches computer programs or scripts at pre-defined times or specific time interval.
https://en.wikipedia.org/wiki/Windows_Task_Scheduler
- Direct Kernel Object manipulation: A common rootkit technique to hide malicious program from task manager and event scheduler.
- E.g. (EProcess structure unlinking)



https://en.wikipedia.org/wiki/Direct_kernel_object_manipulation

- Strings: Command that extracts strings out of files that aren't primarily text-based (guessing it most likely convert bytes values into ascii strings)

<https://linuxopsys.com/strings-command-in-linux>

- Word Boundaries: In regex, this is seen as boundaries between words, denoted as \b and there are three ways to use so for an character/words:

1. Before

2. After

3. Before and after

1. Before (notice how after s there is an word boundary, such as space/tab)

Expression

/s\b/g

Text Tests NEW

This•island•is•beautiful•



2. After (notice how there is an s boundary before the letter i)

Expression

/\bi/g

Text Tests NEW

This•island•is•beautiful•



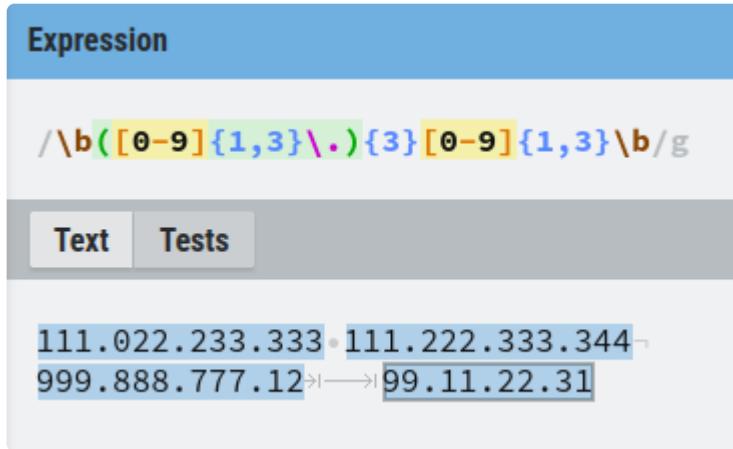
3. Before and after (notice how between the ip address there are boundaries (tab, new line and space))

Expression

/\b([0-9]{1,3}\.){3}[0-9]{1,3}\b/g

Text Tests

111.022.233.333•111.222.333.344•
999.888.777.12→→99.11.22.31



<https://www.regular-expressions.info/wordboundaries.html>

<https://regexr.com/>

- Timestamp: It is where adversaries modify file time attributes to hide new or changes to existing files.
 - In particular, they modify the timestamps of a file (the modify, access, create, and change times), often to mimic files that are in the same folder.
 - Looking at the MFT attribute would reveal the file genuine modification date, as attack need to have kernel level access to tamper that
 - If looking at the different in date between \$FILE_NAME and \$STANDARD_INFO have different values for date then that's a usually sign of timestamping.

- Zone.Identifier: It is a file that describes the security zone associated with another file.
 - It usually describes how "safe" the origin of the file is.
 - It is also an alternative data stream for the file.
 - It is also in an entry of the MFT

<https://fileinfo.com/extension/zone.identifier>

- Logical drive: Defined as the portion of a storage drive, typically defined a letter in front of it, such as C:\.

<https://www.computerhope.com/jargon/l/logidriv.htm>

- Parsing: In forensics, this means taking readable data and organising it into neat, usable format.

<https://cellebrite.com/en/glossary/parsing-mobile-device-forensics/>

- Get-Item vs Get-Content: Get-Item gets you the item while Get-Content gets you the content of the item. It's like getting a bag, Get-Item gets you the bag while Get-Content gets you what's inside the bag (?)

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-content?view=powershell-7.4>

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-item?view=powershell-7.4>

- Non-terminating vs terminating errors: Non-terminating errors are errors that generate a warning to the user but doesn't shutdown the script while terminating errors are those that do

<https://www.oreilly.com/library/view/mastering-windows-powershell/9781787126305/bb1584d6-221a-4b22-8b65-8a2b1cb0e847.xhtml>

- Refer url: The url refers/directs you to the host address (e.g. in an Zone.Identifier file stream) usually through an link.

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer>

- Mark of the Web (Motw): It is a security mechanism (warns user if they want to execute the file and if it's a word doc it runs in restricted mode... etc) that Windows use if files are downloaded from unsafe sources like the Internet. It is implemented through

ADS on zone.identifier of a file.

<https://www.outflank.nl/blog/2020/03/30/mark-of-the-web-from-a-red-teams-perspective/>

- MTFECmd: An command line MFT parser (also some similar files like USN Journal denoted as \$J), particularly designed for NTFS file system.
<https://binaryforay.blogspot.com/2018/06/introducing-mftecmd.html>
<https://github.com/EricZimmerman/MFTECmd>
- .crdownload: A partially downloaded file created by google chrome, ms edge and chromium browsers that stores the content of the file when downloading it.
- Chromium browser: An open source software project first created by google that are used as an "base" or customised by many other browsers including Opera and Microsoft edge.
<https://www.microsoft.com/en-us/edge/learning-center/what-is-chromium-how-does-it-enhance-your-browser?form=MA13I2>
<https://fileinfo.com/extension/crdownload>
- Disable AntiSpyware registry key: By making the value to 1, we set the value of the key to true and turns on Microsoft Defender Antivirus.
<https://learn.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/security-malware-windows-defender-disableantspyware>
- scrape-events.ps1: Useful powerful shell function for parsing Sysmon events from the EQL repository.
<https://www.elastic.co/blog/getting-started-eql>
- EQL query structure on Wildcard: The syntax equivalence as shown below.

wildcard(value, wildcard [, ...])

Compare a value to a list of wildcards. Returns true if any of them match. For example, the following two expressions are equivalent.

```
command_line == "* create *" or command_line == "* config *" or command_line == "* start *"  
wildcard(command_line, "* create *", "* config *", "* start *)
```

-> One thing to note it doesn't go in great depth of the syntax and is different with the eql manual page (no syntax like process where process_name but just uses the function wildcard)

-> Good reference guide overall

<https://eql.readthedocs.io/en/latest/query-guide/functions.html>

- NTUSER.Dat: It is a hidden file that contains setting and preferences for each user.
- UsrClass.Dat: It is a user data file present on all computers
<https://answers.microsoft.com/en-us/windows/forum/all/delete-a-dat-file/5744c250->

105d-460a-bc52-d6c17948764d

<https://www.howtogeek.com/401365/what-is-the-ntuser-file/>

- Registry Transaction log files: These are files that act as journals that stores data written to the registry before written to the hive files, as they may be locked for writing or corruption.

<https://cloud.google.com/blog/topics/threat-intelligence/digging-up-the-past-windows-registry-forensics-revisited/>

- Dirty hives: It is when the hives are probably not the most updated (since windows registry gets updated/flushed once every hour), so we need to replay the transaction log to make it "clean".

<https://medium.com/@mohit.r.phy/exploring-the-hive-deep-inside-the-windows-registry-pt-2-a46d7a3a7f7>

<https://medium.com/@shunxianou/tryhackme-windows-forensics-1-detailed-write-up-5f4e3eaf8bd2>

- Registry plugins: Based on observing its behaviour and description, they seem to be retrieving registry values or doing something related to registry.
- NIC: Known as the network interface card, it is an piece of hardware that allows computer to communicate with other devices on a network.

- It also represents computer on the network and routers and switches uses the MAC address of the NIC card to identify the computer.

<https://www.codecademy.com/resources/blog/network-interface-card/>

- Static/Dynamic Ip address: An Ip address that does not change/change

<https://www.techtarget.com/whatis/definition/static-IP-address#:~:text=A%20static%20IP%20address%20does,their%20ISP%20to%20change%20it>

- DHCP lease time: The amount of time you "leased" or can you use IP address given by the DHCP server.

<https://lazyadmin.nl/home-network/dhcp-lease-time/>

- Bookmark in Registry Explorer: A shortcut for us to access keys/subkeys of the hive.
- sc.exe: Known as Windows service configuration tool and can be used to control a service.

<https://learn.microsoft.com/en-us/windows/win32/services/controlling-a-service-using-sc>

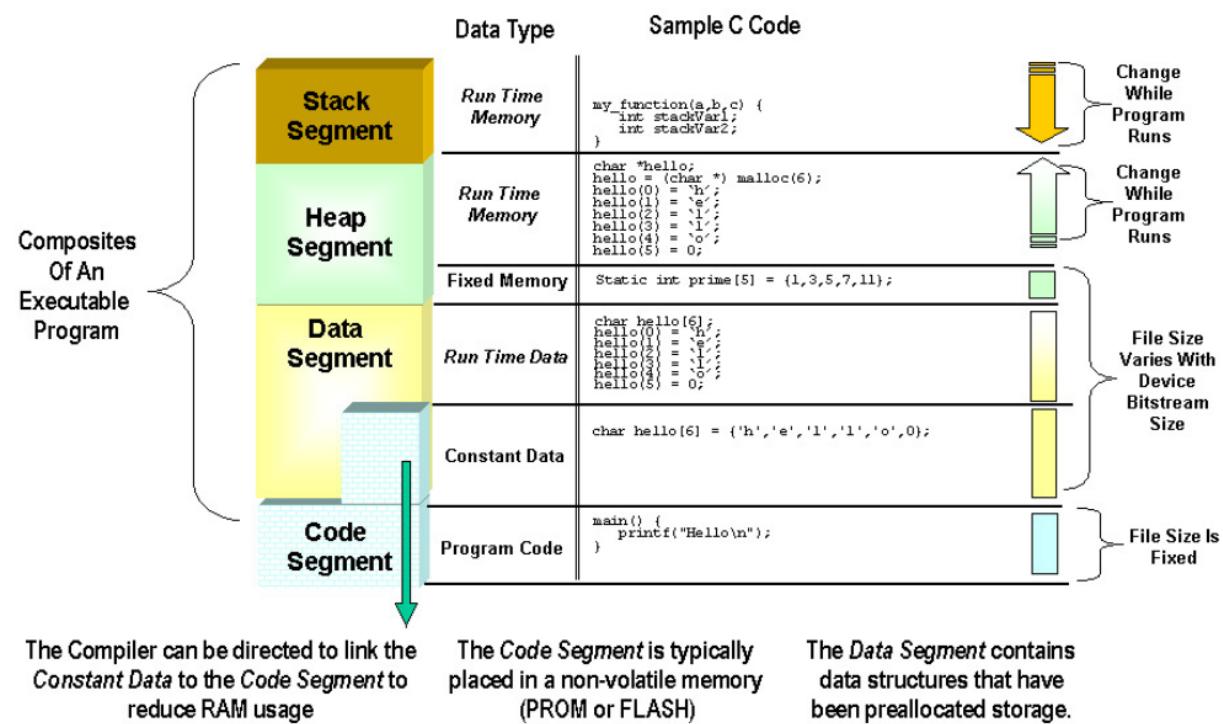
<https://www.file.net/process/sc.exe.html>

- Background Activity Moderator: Known as BAM, it is a windows service that controls activity of background applications.

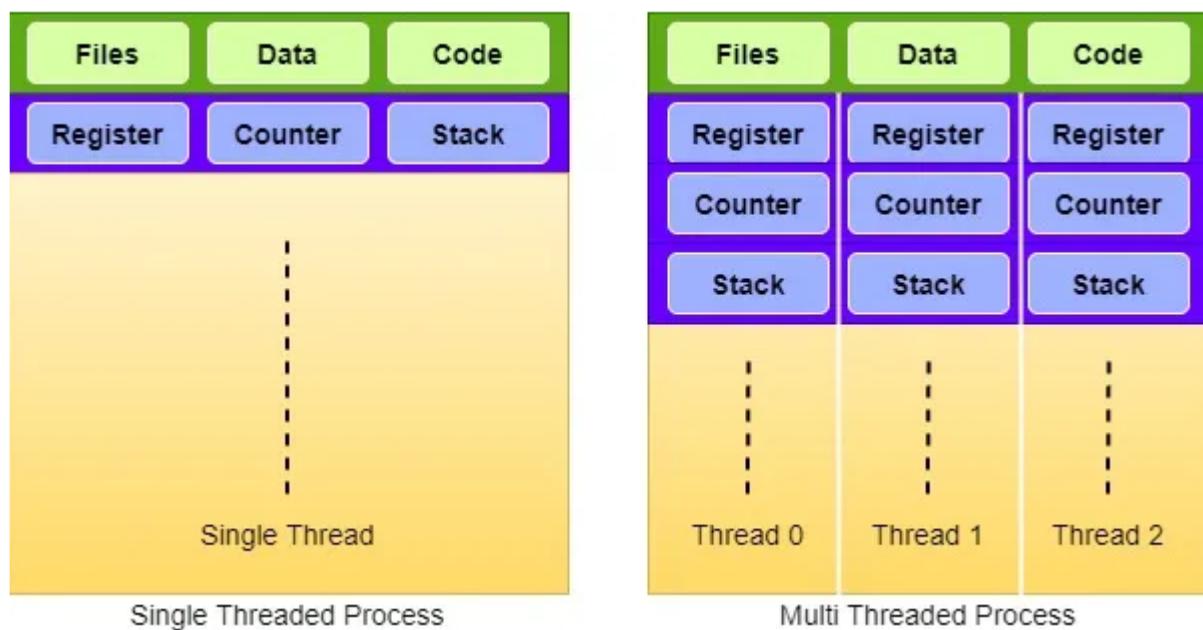
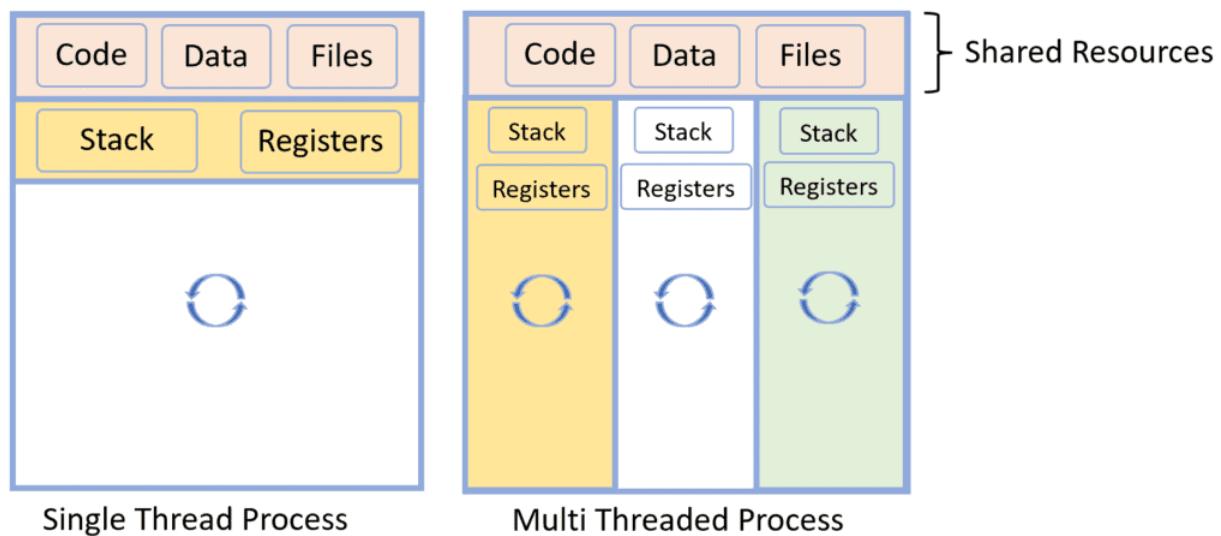
- For forensics, it can be used to provide evidence of program execution.

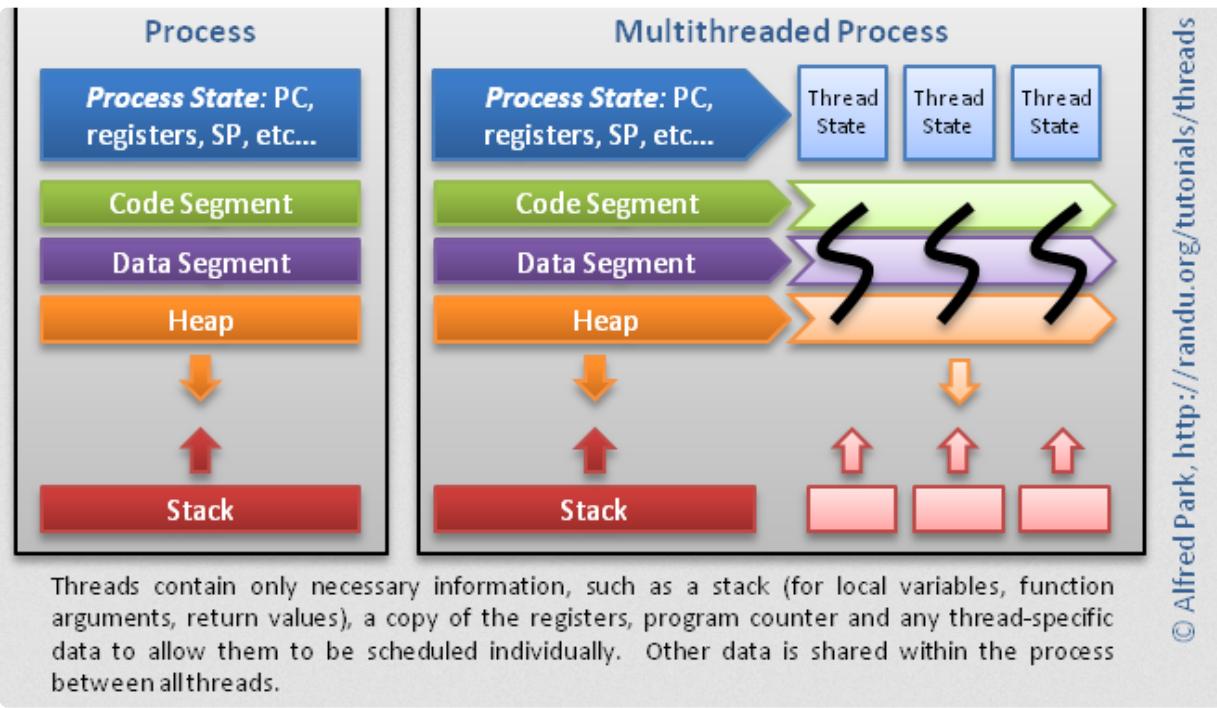
<https://forensafe.com/blogs/bam.html>

- Windows service: A windows service is a program that operates in the background (without user intervention).
https://en.wikipedia.org/wiki/Windows_service
- RegOpenKeyExA: Windows API that opens an specific registry and receives the handle of the desired registry through the phkResult pointer that is inputted in the API call.
- RegSetValueExA: Sets the value of an registry key, which is usually combined with the registry key handle obtained via RegOpenKeyExA.
<https://learn.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regopenkeyexA>
<https://learn.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regsetvalueexA>
- REG_SZ registry value: An null-terminated string that is either a unicode or ANSI string.
<https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry-value-types>
- ANSI, unicode encoding: They are a specific type of encoding, while ANSI a modified form of ISO-8859-1 encoding, a superset of ANSI encoding (the one we commonly see), while Unicode is a encoding standard aimed at Universality.
<https://www.howtogeek.com/45765/htg-explains-what-are-character-encodings-and-how-do-they-differ/>
- Process Injection: A technique where adversaries execute arbitrary code in the address space of a separate live process.
<https://attack.mitre.org/techniques/T1055/>
- Memory segment of a executable program: Stack, heap, data and code segment.
E.g. Visualisation



- Thread: An execution context for a process (book analogy: threads of the same program are like bookmarks of the same book, while CPU is like the person reading it).
 - The main difference between thread and processes are that distinct process exist in different memory spaces while distinct threads of a same process share all segment of a executable program except the stack.
 - Moreover, though threads have independent stack, we can still have pointers toward it and access the other thread's local memory.
 - E.g. Example of threads





<https://stackoverflow.com/questions/5201852/what-is-a-thread-really>

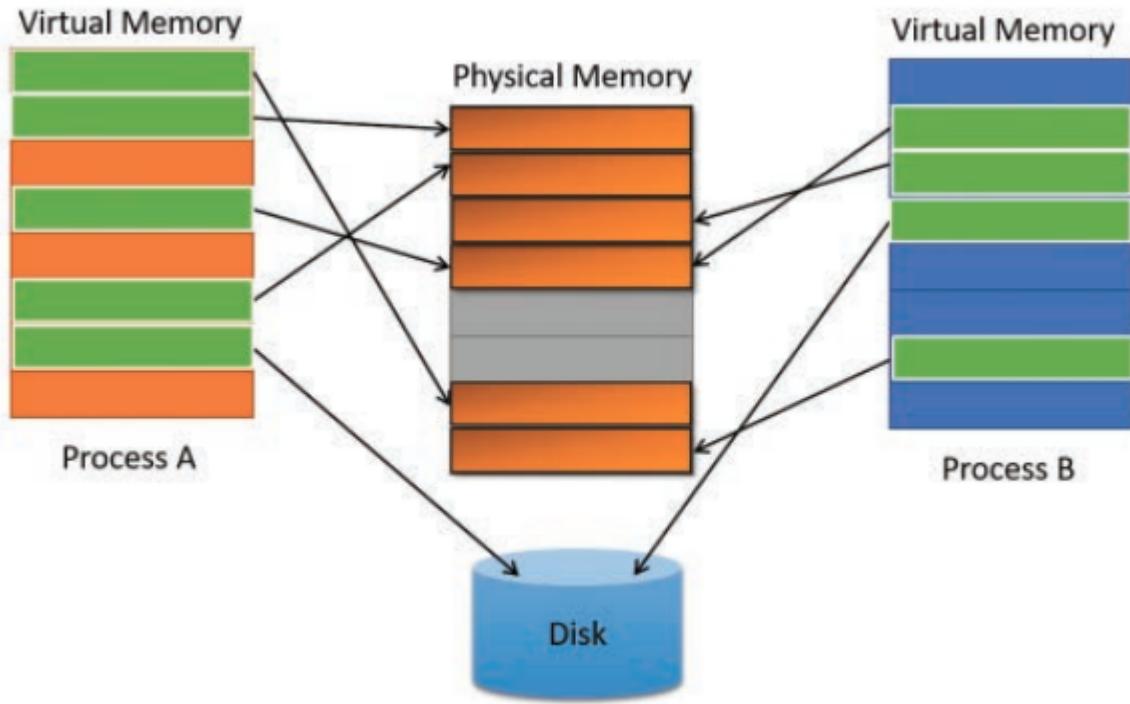
<https://stackoverflow.com/questions/1762418/what-resources-are-shared-between-threads>

<https://www.baeldung.com/cs/threads-sharing-resources>

<https://medium.com/@jithmisha/a-brief-intro-to-shared-memory-programming-with-posix-threads-a663b590e38c>

- CreateProcessA: A windows api function that creates a new process and its primary thread.
<https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa>
- Create_Suspended process creation flag: It makes the primary thread of the new process in a suspended state and does not run until ResumeThread function is called.
<https://learn.microsoft.com/en-us/windows/win32/procthread/process-creation-flags>
- Memory paging: It is an windows memory management mechanism that divides memory into chunks of 4kb called "pages".
 - Memory in windows OS are also not directly mapped to physical memory, instead they use a virtual memory that is mapped to physical memory.
 - The end goal seems to be saving as much physical memory as possible and allows processes to have different virtual memory but share the physical memory.

E.g Illustration from System internals



- Memory page state: The state of the virtual memory within a process can be in one of three states:
 - Free: The memory can be reserved, committed, or simultaneously reserved and committed.
 - Reserved: It is where the page has been reserved for future use and this reserved addresses can't be used by other memory allocation function such as malloc.
 - Committed: This is where memory charges has been made from the overall size of RAM and paging files on the disk (i.e. mapping between virtual and physical memory has been achieved).
 - Three access to committed pages is also determined by the memory protection constant set on the page (e.g. includes page_no_access, page_execute_read_write, page_readonly which their name shows what type of access is allowed on the page).
- OpenProcess: Windows API that opens the processes and returns an handle to the open process
- VirtualAlloc: Windows API that allocates memory (reserve, commit or change) given a handle to a process and returns a handle of the base address of the allocated region of pages.
- WriteProcessMemory: Write data to an specified area of memory in a specified process (given the process handle, base address of memory to write, data to write and size of data to write) which returns a non-zero value if it succeeds.
- CreateRemoteThread: Creates a thread that runs in the virtual address of another process (given the process handle, base address of thread execution) and returns a

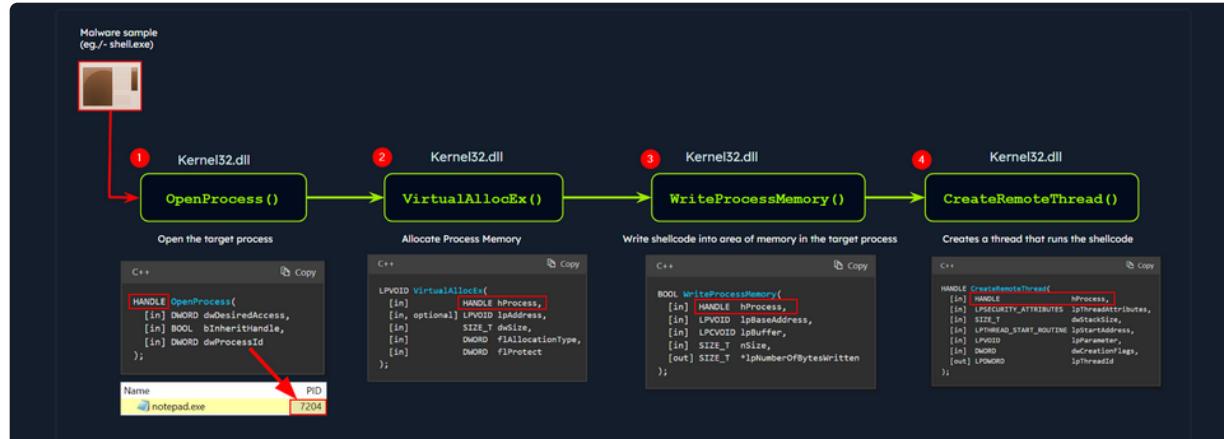
handle to the thread if succeed.

<https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess>

<https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex>

<https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory>

- E.g. From malware analysis for a succinct summary.



In this diagram, the malware process (`shell.exe`) performs process injection to inject code into a target process (`notepad.exe`) using the following functions imported from the DLL `kernel32.exe`:

- OpenProcess**: Opens a handle to the target process (`notepad.exe`), providing the necessary access rights to manipulate its memory.
- VirtualAllocEx**: Allocates a block of memory within the address space of the target process to store the injected code.
- WriteProcessMemory**: Writes the desired code into the allocated memory block of the target process.
- CreateRemoteThread**: Creates a new thread within the target process, specifying the entry point of the injected code as the starting point.

- Security_Attributes structure: A structure that contains the security descriptor for an object and specifies whether this structure is inheritable.

[https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa379560\(v=vs.85\)](https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa379560(v=vs.85))

- Signed script: Powershell script are signed when they have a digital signature (encrypted message using author's private key), which we can choose to trust or (through validating the digital certificate of the author with the chain of CA).

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_signing?view=powershell-7.4

- Powershell execution policy: A safety feature that controls the condition under which Powershell load configuration files and run script.

- It is important to note that it is not a security feature as execution policy can easily be

bypassed.

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-7.4

- Schtasks: A Windows command that schedules tasks, including the creation of tasks.
<https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/schtasks-create>
- Rundll32: This program is also known as "Run a dll as an App" program can execute dll's files, which malware can use to execute malicious code.
<https://malwaretips.com/blogs/rundll32-exe-what-it-is-should-i-remove-it/>
- StartW: In Rundll32, this is function that executes the dll present. It is often associated with Cobalt Strike payload.
<https://www.microsoft.com/en-us/security/blog/2022/04/13/dismantling-zloader-how-malicious-ads-led-to-disabled-security-tools-and-ransomware/>
<https://research.splunk.com/endpoint/9319dda5-73f2-4d43-a85a-67ce961bddb7/>
- Autopsy case: Like the name suggests, it is the "case" that we are dealing with. In the case we can add data sources and do a lot more things. To open a case we select the .aut file that already exists.
https://sleuthkit.org/autopsy/docs/user-docs/3.1/cases_page.html
<https://www.youtube.com/watch?v=fEqx0MeCCHg>
- Chrome Cache file: A file format used by Google Chrome and Chromium to store persistent cache data.
 - Combining insights gained from Chrome Cache file and the location of the file we obtain, we may be able to obtain further insights (e.g. file probably downloaded from internet, as it appears in download folder + cache file).
<https://github.com/libyal/dtformats/blob/main/documentation/Chrome%20Cache%20file%20format.asciidoc>
- .arn: Known as Native auto run binary file format, it displays everything that runs automatically when Windows starts. It can be viewed using applications like Autoruns.
- Publisher verification: A mechanism implemented by Microsoft to give users about the authenticity developer's platform.
 - Not-verified means the developer's platform of the app can be suspicious.
<https://learn.microsoft.com/en-us/entra/identity-platform/publisher-verification-overview>
- Svchost.exe: Known as the service host, it serves as a shell for loading services from DLL files.
 - It is also known as malware impersonating it due to the relative ease of doing so (name the malware svchost.exe and give its description of "Host Process for Windows Services" and it would be hard to distinguish it from the legitimate svchost.exe)
 - Some notable flags include -k and -s flag.
 - When it uses the -k flag, it will look for the key at an registry key name, located at

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows

NT\CurrentVersion\Svchost which has references to various service name (which it can load).

- E.g. See the shell command below and it looking it at the registry key name UnistackSvcGroup at the location above.

- When it uses the -s flag, it is usually combined with the -k flag and will tell svchost.exe to load the specific service name.
 - E.g. (Load only UnistackSvcGroup with combo of svchost shell command)

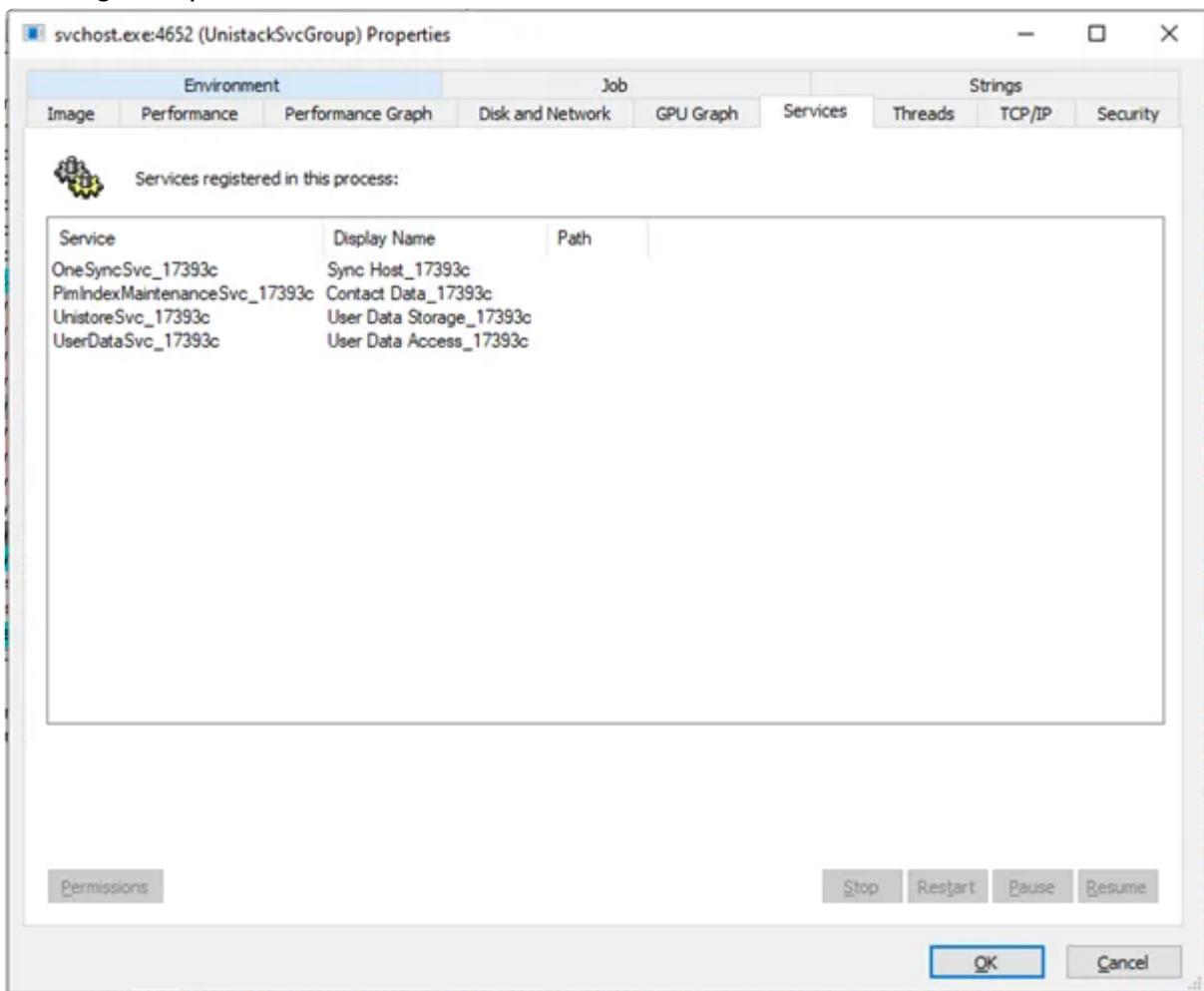
![[Pasted image 20240812135957.png]]

E.g. Svchost shell command and it hosting services:

Shell Command



Hosting multiple services:



- Svchost.exe also commonly run with System privileges when being abused by malware.

How Malware Abuses SvcHost.exe

Cybercriminals often target svchost.exe for malware impersonation because of its common presence and high privileges. Tactics include:

1. Mimicry

Malware authors name executables svchost.exe and install them in the system folders. This makes them appear legitimate and blend into the crowd of real svchost entries.

2. Service Hijacking

Some malware hijacks existing svchost-hosted services by registering themselves as a service. This grants them the same trusted execution context as critical system services.

4. Privilege Escalation

Svchost runs with SYSTEM privileges, enabling full access to the OS. By impersonating it, malware inherits powerful permissions to bypass restrictions.

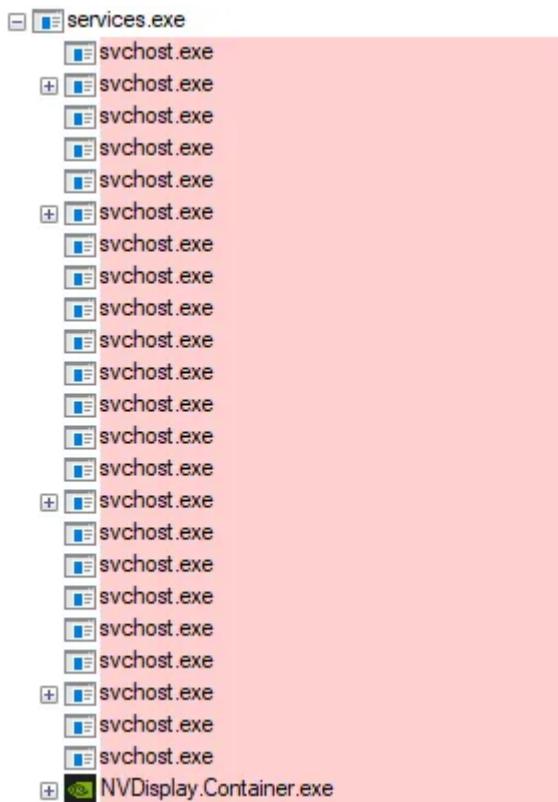
5. Stealth

Blending in as one of numerous svchost instances allows malware to operate discreetly, circumventing detection. Malware svchosts are sometimes hidden entirely from task managers.

https://malwaredetect.com/blogs/svchost-exe-virus-removal/#normal_behavior_for_svchostexe

- Service.exe: It is known as the process for running and managing services on the system.

- We'd often seen svchost.exe as an children of Service.exe, as shown below:



<https://nasbench.medium.com/demystifying-the-svchost-exe-process-and-its-command-line-options-508e9114e747>

- User Account Control: A mechanism that prompts the user for credential or approval when it performs operations or tries to access app at an elevated level, as the app requires the user's administrative token (Windows by default let admin access resources and execute apps under the security context of a standard user).

- UAC can be bypassed if some applications run in auto-elevated context, such as fodhelp.exe with autoelevated feature set to true.
- It often just makes life harder for attacker's and generate noise and is not an security measure.

<https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/how-it-works>

<https://pentestlab.blog/2017/06/07/uac-bypass-fodhelper/>

- ParseExact: It converts a string representation of time into its DateTime object struct equivalent (which represents an instant in time).

<https://stackoverflow.com/questions/50214955/how-to-parse-datetime-by-parseexact-in-powershell>

<https://learn.microsoft.com/en-us/dotnet/api/system.datetime.parseexact?view=net-8.0>
<https://learn.microsoft.com/en-us/dotnet/api/system.datetime?view=net-8.0>

- Import-CSV: Creates a table-like custom object for items in a CSV file.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/import-csv?view=powershell-7.4>

- Where-Object: Selects object from a collection (collection of objects) based on their property value.

- `$_` is an automatic variable which represents each object that is passed to the where-Object cmdlet.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/where-object?view=powershell-7.4>

- -as: In powershell this is like object casting (based on behaviour observed)

E.g. Powershell (casting object timestamp as `[DateTime]` object)

```
Where-Object { $_.timestamp -as [DateTime] -ge $time1 -and $_.timestamp -as [DateTime] -lt $time2 }
```

- USN Journal Reason: In USN journal, this is the flag that identify reasons for changes that have accumulated in this file/directory record since the directory opened.

https://learn.microsoft.com/en-us/windows/win32/api/winioclt/ns-winioclt-usn_record_v2

- Select-String: Finds text in strings and files

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/select-string?view=powershell-7.4>

- Pagefile.sys: It is a file that uses physical memory to help supplement the system's RAM when it is almost full, which offloads some less critical file to it.

- Being a physical memory, it will be slower to read from it than RAM, but faster than closing the app than reopening it.

- It is usually (by default) located in the C:\ drive

<https://www.makeuseof.com/windows-pagefile-sys-guide/>

- Plaso: An python-based engine used by several tools for the automatic creation of timelines.

- It is also what Autopsy use behind the scenes.

<https://github.com/log2timeline/plaso>

- Sans triage parameter: An configuration parameter for KAPE which allows useful collection of artifacts for KAPE.

- E.g. What Sans triage collects (can see useful files like usn journal, mft file... etc)

```
_SANS_Triage     SANS Triage Collection (by Mark Hallman): $Boot, $J, $J, $LogFile, $MFT, $Max, $Max, $SDS, $SDS, $T, $T, AVG AV Logs, AVG AV Logs (XP), AVG AV Report Logs (XP), AVG FileInfo DB, AVG Persistent Logs, AVG Report Logs, AVG lsdbj2 JSON, ActivitiesCache.db, Addons, Addons XP, Amcache, Amcache, Amcache transaction files, Amcache transaction files, Ammyy Program Data, AnyDesk Chat Logs - User Profile, AnyDesk Logs - ProgramData - *.conf, AnyDesk Logs - ProgramData - *.trace, AnyDesk Logs - ProgramData - connection_trace.txt, AnyDesk Logs - System User Account, AnyDesk Logs - User Profile - *.conf, AnyDesk Logs - User Profile - *.trace, AnyDesk Logs - User Profile - connection_trace.txt, AnyDesk Videos, AppCompat PCA Folder, Application Event Log Win7+, Application Event Log Win7+, Application Event Log XP, Application Event Log XP, Avast AV Index, Avast AV Logs, Avast AV Logs (XP), Avast AV User Logs, Avast Icarus Logs, Avast Persistent Data Logs, Avira Activity Logs, Avira Security Logs, Avira VPN Logs, BITS files, Bitdefender Endpoint Security Logs, Bitdefender Internet Security Logs, Bitdefender SQLite DB Files, Bookmarks, Bookmarks, Bookmarks, Box Drive Application Metadata, Box Sync Application Metadata, Chrome Cookies, Chrome Cookies XP, Chrome Current Session, Chrome Current Session XP, Chrome Current Tabs, Chrome Current Tabs XP, Chrome Download Metadata, Chrome Extension Cookies, Chrome Favicons, Chrome Favicons XP, Chrome History, Chrome History XP, Chrome Last Session, Chrome Last Session XP, Chrome Last Tabs, Chrome Last Tabs XP, Chrome Login Data,
```

- Virtual Address Descriptors (VADs): It is a set of data structures that each process maintains, where it each of them describes the range of virtual address in the process's address space have been reserved or not.

- These set that each process maintain are stored in a self-balanced binary tree for efficiency purpose.

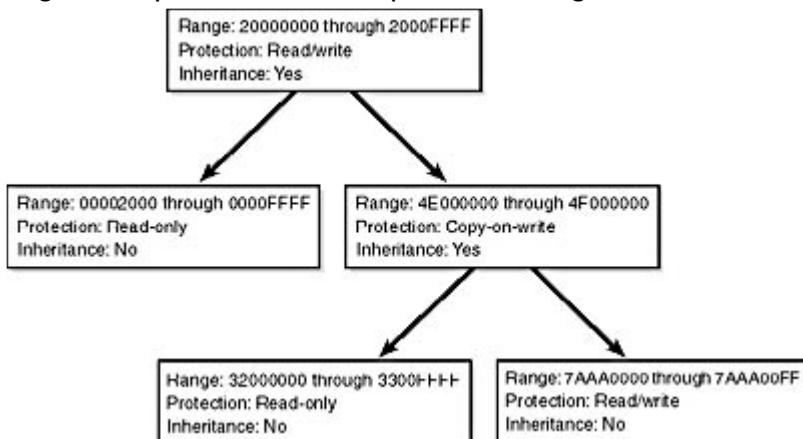
- They are created because memory manager uses a demand paging in the OS, so it

needs to know the status of each memory range (you can load memory in physical memory later, but you must have a mechanism of tracking it before it gets loaded).

- Each node also has some useful flags associated with them, these include the following:

- Protection: This field gives an indication of the type of access allowed to the memory region. Some of the protection constants are:
 - PAGE_EXECUTE: Memory can be executed but cannot be written to
 - PAGE_EXECUTE_READ: Memory can be executed or read but cannot be written to
 - PAGE_EXECUTE_READWRITE: Memory can be executed, read and write.
 - PAGE_NOACCESS: No access to this memory region
 - PAGE_READONLY: Only read access to the memory
 - PAGE_READWRITE: Read, Write access to the memory but no execution.
- Private Memory: This field refers to committed regions that cannot be shared with other processes.

E.g. Example of VADs in a process using self-balanced binary tree



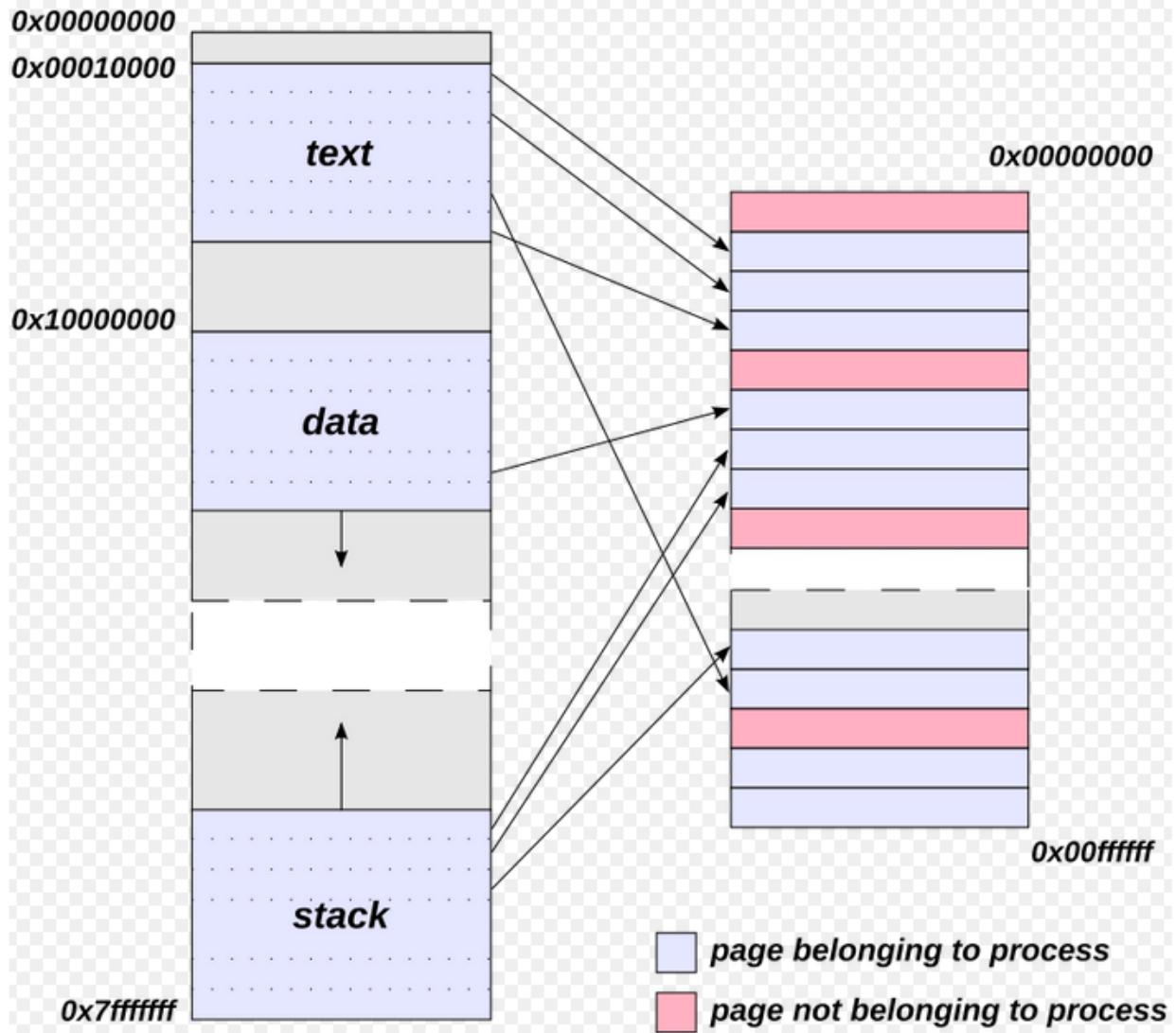
<https://flylib.com/books/en/3.169.1.64/1/>

<https://www.infosecinstitute.com/resources/penetration-testing/finding-enumerating-processes-within-memory-part-2/>

- Demanding Paging: A memory management scheme used in OS that loads memory in a "lazy evaluation" fashion (load into memory only when needed through page fault mechanism)
 - This improves the performance of OS and system performance.
- Page Fault: A technique used in virtual memory systems where page enters main memory (actual mapping achieved) only when requested or needed by the CPU.
 - A page fault occurs when the program needed to access a page that is not currently in memory, which the OS then loads the required page from disk into memory and update the tables (mapping table between virtual and physical memory) accordingly.
 - E.g. Page table for virtual address translation

Virtual address space

Physical address space



<https://www.geeksforgeeks.org/what-is-demand-paging-in-operating-system/>

https://en.wikipedia.org/wiki/Page_table

Evidence Acquisition Techniques & Tools

Question

- Visit the URL "<https://127.0.0.1:8889/app/index.html#/search/all>" and log in using the credentials: admin/password. After logging in, click on the circular symbol adjacent to "Client ID". Subsequently, select the displayed "Client ID" and click on "Collected". Initiate a new collection and gather artifacts labeled as "Windows.KapeFiles.Targets" using the _SANS_Triage configuration. Lastly, examine the collected artifacts and enter the name of the scheduled task that begins with 'A' and concludes with 'g' as your answer.

-> We login and start collect the artifacts using kapefiles.target and download as follows

The screenshot shows a software interface for managing hunts. At the top, there's a toolbar with icons for adding, deleting, and filtering. Below it is a table listing three hunts:

State	Hunt ID	Description	Created	Started	Expires	Scheduled	Creator
X	H.CQ03UOASACPOQ	teste hunt 2	2024-08-05T02:56:33Z	2024-08-05T02:56:33Z	2024-08-12T02:53:34Z	1	admin
X	H.CQ03NGU70LQFM	Test hunt	2024-08-05T02:41:07Z	2024-08-05T02:41:07Z	2024-08-12T02:38:28Z	1	admin
■	H.CJ0H7UEPS5CUK	Users	2023-08-31T22:28:09Z	2023-08-31T22:28:09Z	2023-09-07T22:27:00Z	102	admin

On the left, a sidebar displays the parameters for the selected hunt (H.CQ03NGU70LQFM):

- Artifact Names: Windows.KapeFiles.Targets
- Hunt ID: H.CQ03NGU70LQFM
- Creator: admin
- Creation Time: 2024-08-05T02:41:07Z
- Expiry Time: 2024-08-12T02:38:28Z
- State: RUNNING
- Ops/Sec: Unlimited
- CPU Limit: Unlimited
- IOPS Limit: Unlimited
- Parameters:
 - Windows.KapeFiles.Targets
 - _SANS_Triage: Y

On the right, the "Download Results" section shows the following details:

- Total scheduled: 1
- Finished clients: 1
- Available Downloads:
 - H.CQ03NGU70LQFM
- File details for H.CQ03NGU70LQFM:
 - Uncompressed: 1319 Mb
 - Compressed: 263 Mb
 - Container Files: 1355
 - Started: 2024-08-05T02:43:54Z
 - Duration (Sec): 40
 - SHA256 Hash: 0558780afa19f8208f2c17926ab695338e1f7a36cc2b6940bf8d009588768b2c

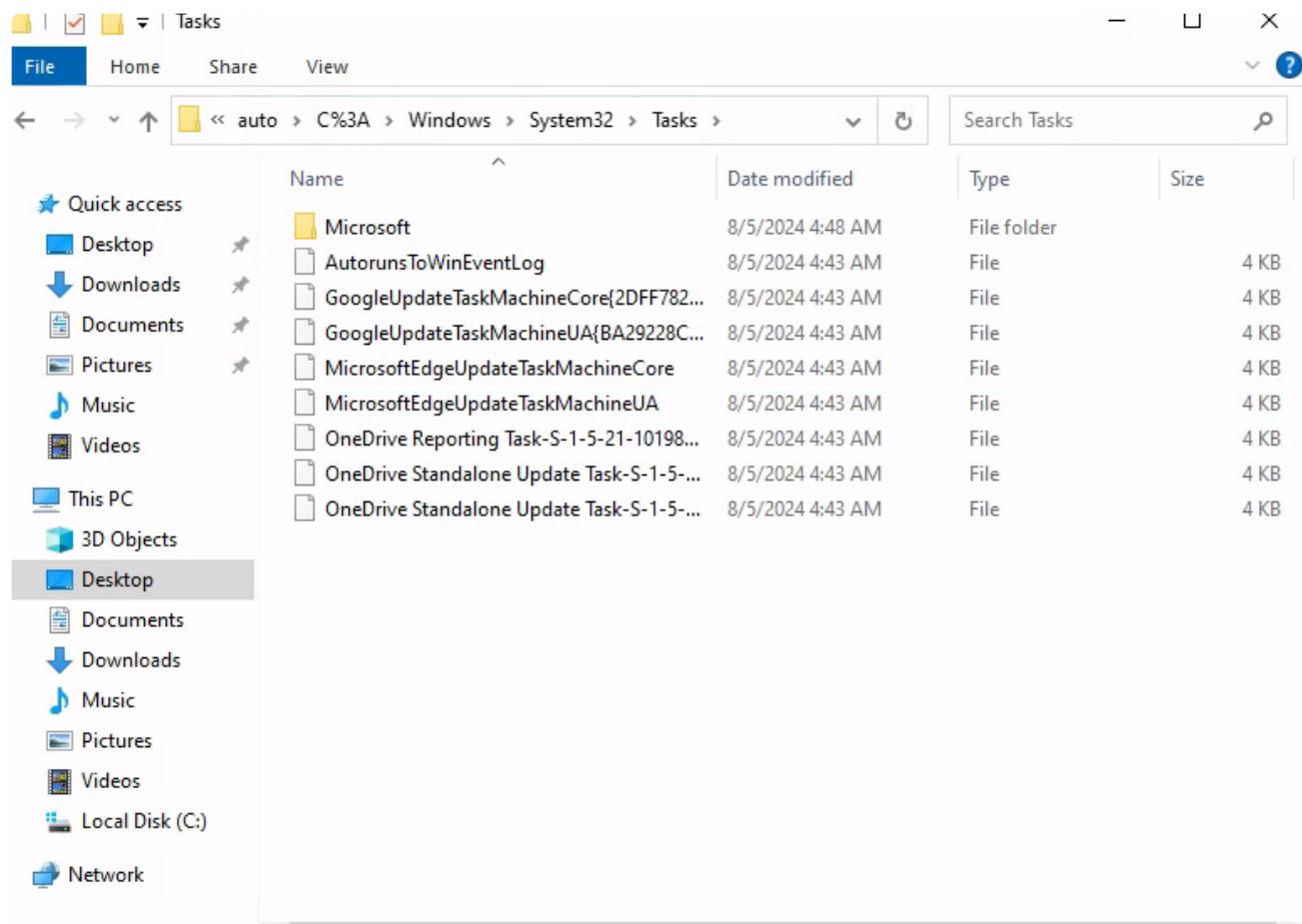
-> We then search up on where the folder for scheduled tasks is located on windows 10 (since our machine is windows 10), where we came across that it is saved on the folder below

C:\Windows\System32\Tasks

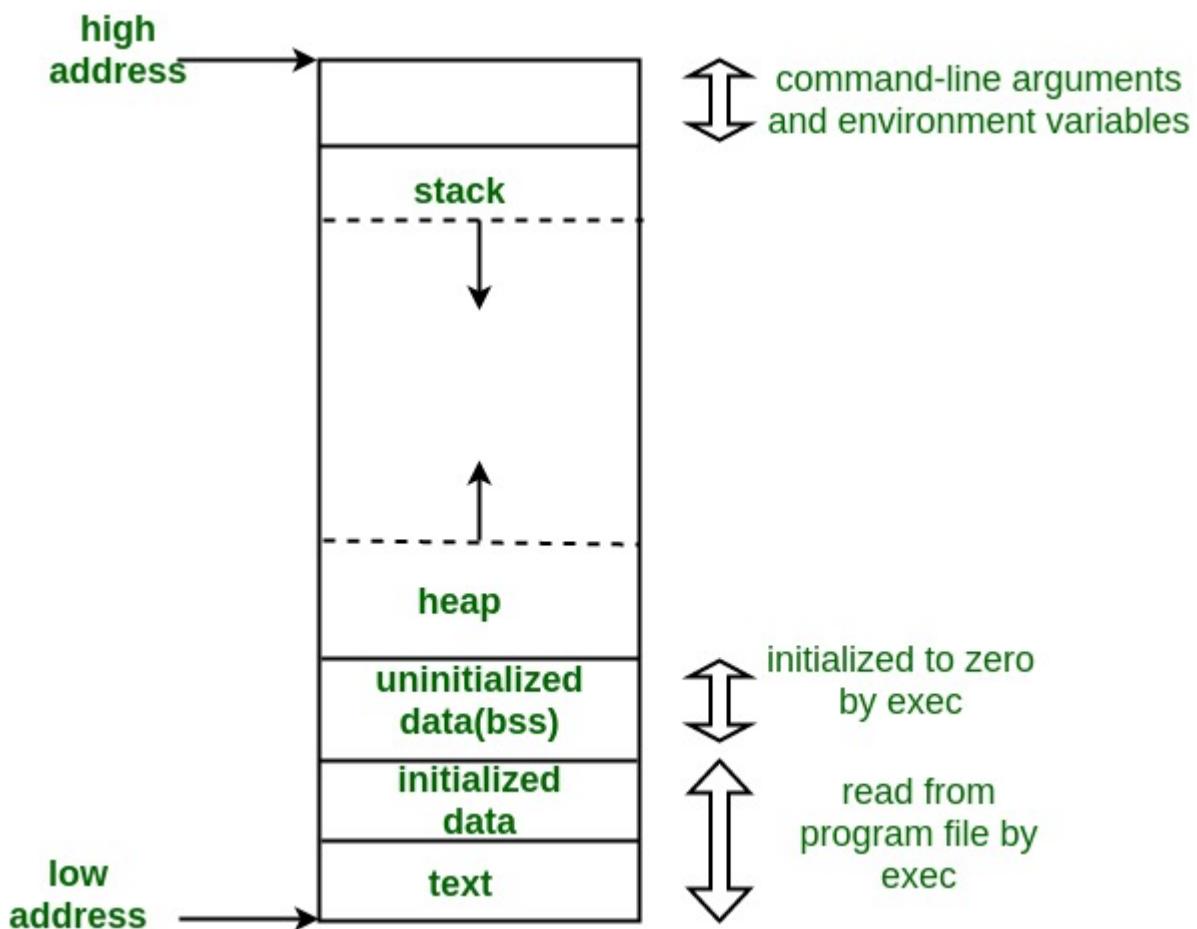
with reference to this stackoverflow link

<https://stackoverflow.com/questions/2913816/how-to-find-the-location-of-the-scheduled-tasks-folder>

-> Then, we go into thhis folder as follows and see that the name of the schedoled tasks is AutorunsToWinEventLog :



- VAD: An artifact available in Velociraptor that allows enumeration of process memory sections via Virtual Address Descriptor (VAD).
https://docs.velociraptor.app/artifact_references/pages/windows.system.vad/
- Process Memory section: The sections of the process memory includes code segment, data segment, stack and memory sections.



<https://www.geeksforgeeks.org/memory-layout-of-c-program/>

- Process Environment Block: Known as PEB, it is a process's user-mode representation.
 - It has highest level knowledge of a process in kernel model and lowest-level in user mode.

<https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/api/pebteb/peb/index.htm>
- Impersonation token: An impersonation access token describes the security context of the client being impersonated.

<https://learn.microsoft.com/en-us/windows/win32/secauthz/impersonation-tokens>
- Hard link: A file-representation of a file through which more than one path references the same file in the same volume.
 - Basically hard copies of file that are linked (changing file in a hard link reflects changes to files in all other hard link) in the same volume.

<https://learn.microsoft.com/en-us/windows/win32/fileio/hard-links-and-junctions#hard-links>

Evidence Examination & Analysis

Memory Forensics

Question

- Examine the file "/home/htb-student/MemoryDumps/Win7-2515534d.vmem" with Volatility. Enter the parent process name for @WanaDecryptor (Pid 1060) as your answer. Answer format: _.exe

-> We first examine the OS profile of the memory dump:

```
vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem imageinfo
```

```
htb-student@remnux:~$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem imageinfo
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
    from cryptography.hazmat.backends.openssl import backend
INFO    : volatility.debug    : Determining profile based on KDBG search...
    Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000, Win7SP1x64_23418
AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/home/htb-student/MemoryDumps/Win7-2515534d.vmem)
PAE type : No PAE
DTB : 0x187000L
KDBG : 0xf80002be9120L
Number of Processors : 1
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0xfffff80002beb000L
KUSER_SHARED_DATA : 0xfffff78000000000L
Image date and time : 2023-06-22 12:34:03 UTC+0000
Image local date and time : 2023-06-22 18:04:03 +0530
```

-> Seems like the profile is Win7SP1x64 `Win7SP1x64`

-> Now we perform pslist to analyse the process id and it's associated parent:

```
vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --
profile=Win7SP1x64 pslist
```

```
htb-student@remnux:~$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
    from cryptography.hazmat.backends.openssl import backend
Offset(V)      Name          PID  PPID  Thds  Hnds  Sess  Wow64 Start          Exit
0xffffffffa8001d22b00 tasksche.exe      1792  1044    8     82    0    1 2023-06-22 12:31:13 UTC+0000
0xffffffffa8002fa3060 SearchProtocol   852   2756    8    289    0    0 2023-06-22 12:31:15 UTC+0000
0xffffffffa8002572060 @WanaDecryptor 1060   1792    2     71    0    1 2023-06-22 12:31:27 UTC+0000
```

-> Hence, the parent process name is tasksche.exe

-> Or, we can use pstree for a more clearer illustration of parent-child relationship (reference to github wiki page)

```
vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 pstree
```

```
htb-student@remnux:~$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 pstree
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2
is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release
...
from cryptography.hazmat.backends.openssl import backend
Name      Pid  PPid  Thds  Hnds  Time
0xfffffa8001d22b00:tasksche.exe          1792  1044     8    82  2023-06-22 12:31:13 UTC+0000
. 0xfffffa8002572060:@WanaDecryptor       1060  1792     2    71  2023-06-22 12:31:27 UTC+0000
```

- Examine the file "/home/htb-student/MemoryDumps/Win7-2515534d.vmem" with Volatility. tasksche.exe (Pid 1792) has multiple file handles open. Enter the name of the suspicious-looking file that ends with .WNCRYT as your answer. Answer format: ___.WNCRYT

-> We examine the handle for this process, specific to the file handle

```
vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handles -p 1792 --object-type=File
```

```
htb-student@remnux:~$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handles -p 1792 --object-type=File
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2
is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release
...
from cryptography.hazmat.backends.openssl import backend
Offset(V)      Pid  Handle with Volatility, ta Access Type 1792) has multip Detailsles open. Enter the
...
0xfffffa8000ea8f20  1792  0x10  Answer format: \Device\HarddiskVolume2\Windows
0xfffffa8002d5b8c0  1792  0x1c  0x100001 File        \Device\KsecDD
0xfffffa8000e2e070  1792  0x5c  We examine the 0x100000 File process, specific \Device\HarddiskVolume2\ProgramData\ggzstcat367
0xfffffa8002ca7390  1792  0x64  0x100001 File        \Device\KsecDD
0xfffffa8011bf0d070 1792  0xf8  0x120196 File        \Device\HarddiskVolume2\ProgramData\ggzstcat367\00000000
.eky               1792
0xfffffa8001e1e070  1792  0x148 0x120196 File        \Device\HarddiskVolume2\Windows\Temp\hibsys.WNCRYT
```

-> We can see the suspicious-looking file is hibsys.WNCRYT

- Examine the file "/home/htb-student/MemoryDumps/Win7-2515534d.vmem" with Volatility. Enter the Pid of the process that loaded zlib1.dll as your answer.
-> We look at the dll's loaded for all process (with reference to github wiki page)

```
vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 dlllist
```

-> That's a lot of info displayed, so we will print 50 lines before grepping for zlib1.dll

```
vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 dlllist | grep "zlib1.dll" -B 50
```

base	size	path
0x00000000001230000	0x2fe000	vol.py 0xffff 1970-01-01 00:00:00 UTC+0000 C:\ProgramData\ggzstcat367\TaskData\Tor\taskhsvc.exe
0x0000000000773f0000	0x19f000	profile=Win7SP1x64 dlllist
0x0000000000739d0000	0x3f000	0xffff 1970-01-01 00:00:00 UTC+0000 C:\Windows\SYSTEM32\ntdll.dll
0x000000000073970000	0x5c000	→ That's a lot of info displayed, so we will print 50 lines before grepping for zlib1.dll
0x000000000073960000	0x8000	0x3 2023-06-22 12:31:29 UTC+0000 C:\Windows\SYSTEM32\wow64.dll
0x00000000001230000	0x2fe000	vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 dlllist grep "zlib1.dll" -B 50
0x000000000075b0000	0x180000	0x1 2023-06-22 12:31:29 UTC+0000 C:\Windows\SYSTEM32\wow64win.dll
0x000000000075b50000	0x110000	0x1 2023-06-22 12:31:29 UTC+0000 C:\Windows\SYSTEM32\wow64cpu.dll
0x0000000000770c0000	0x47000	0xffff 1970-01-01 00:00:00 UTC+0000 C:\ProgramData\ggzstcat367\TaskData\Tor\taskhsvc.exe
0x00000000006b630000	0x82000	0xffff 1970-01-01 00:00:00 UTC+0000 C:\Windows\SysWOW64\ntdll.dll
0x00000000006b610000	0x1c000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\kernel32.dll
0x000000000074d30000	0xa1000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\KERNELBASE.dll
0x000000000077110000	0xac000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\msvcr1.dll
0x000000000076450000	0xc4000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\SHLWAPI.dll
0x000000000075c50000	0x57000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\GDI32.dll
0x000000000074ee0000	0x90000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\USER32.dll
0x000000000075e60000	0x100000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\LPK.dll
0x0000000000770a0000	0xa000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\USP10.dll
0x0000000000750c0000	0x9d000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\WS2_32.dll
0x0000000000755f0000	0x35000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\NSI.dll
0x000000000074f70000	0x6000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\Windows\SysWOW64\LIBEAY32.dll
0x00000000006b370000	0x21c000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\ProgramData\ggzstcat367\TaskData\Tor\SSLEAY32.dll
0x00000000006b2e0000	0x82000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\ProgramData\ggzstcat367\TaskData\Tor\zlib1.dll
0x00000000006b2b0000	0x22000	0xffff 2023-06-22 12:31:29 UTC+0000 C:\ProgramData\ggzstcat367\TaskData\Tor\zlib1.dll

-> Hence, we see it is taskhsvc.exe with pid of 3012 loading the dll.

Rapid Triage Examination & Analysis Tools

Questions

- During our examination of the USN Journal within Timeline Explorer, we observed "uninstall.exe". The attacker subsequently renamed this file. Use Zone.Identifier information to determine its new name and enter it as your answer.
 - > Given that file is renamed, the metadata of the file is not erased, but duplicated over instead.
 - > Hence, we first obtain the zone.identifier information for "uninstall.exe" and use that

to identify the new file name.

-> Given the entry number is given as 93866, and zone identifier is

Entry Number	File Name	Zone Id Contents	Extension	Is Dir
93866	uninstall.exe	[ZoneTransfer] ZoneId=3 RefererUrl=http://10.10.10.10:443/ HostUrl=http://10.10.10.10:443/uninstall.exe	.exe	
	93866 uninstall.exe:Zone.Identifier		.Identifier	

shown in the section, we can use the time explorer to filter for zone id that has the same zone id contents (we search for uninstall in Zone Id Contents)

Drag a column header here to group by that column

ss0x10	Last Access0x30	Zone Id Contents
7 08:30:08	2023-09-07 08:30:06	[ZoneTransfer] ZoneId=3 ReferrerUrl=http://10.10.10.10:443/ HostUrl=http://10.10.10.10:443/uninstall.exe
7 08:29:30	2023-09-07 08:29:11	[ZoneTransfer] ZoneId=3 ReferrerUrl=http://10.10.10.10:443/ HostUrl=http://10.10.10.10:443/uninstall.exe

Zone Id Contents Contains uninstall

mn header here to group by that column Enter text to search

Parent Path	File Name	Extension
\Windows\Tasks	microsoft.windowskits.feedback.exe:Zone.Identifier	.Identifier
\Temp	uninstall.exe:Zone.Identifier	.Identifier

-> We see a file that has the same zone Id Contents, which looks suspicious and is last accessed later than the uninstall.exe (bottom one)

->We obtain the entry number of the file and look into it using MTFECmd.exe:

Line	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	In Use	Parent Path
=	<input checked="" type="checkbox"/>	=	=	=	=	<input type="checkbox"/>	Root
115697	<input type="checkbox"/>	90469	4	4241		<input checked="" type="checkbox"/>	.\Windows\Tasks
120500	<input type="checkbox"/>	93866	3	92487		<input type="checkbox"/>	.\Temp

```
.\MFTECmd.exe -f  
'C:\Users\johndoe\Desktop\forensic_data\kapē_output\%MFT%' --de 90469
```

-> Hence, we obtain that the new file name is `microsoft.windowskits.feedback.exe`

- Review the file at "C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%4Operational.evtx" using Timeline Explorer. It documents the creation of two scheduled tasks. Enter the name of the scheduled task that begins with "M" and concludes with "r" as your answer.
- > We first do the conversion from .evtx file to .csv file

```
.\\EvtxECmd.exe -f
'C:\\Users\\johndoe\\Desktop\\forensic_data\\cape_output\\Windows\\System32\\winevt\\logs\\Microsoft-Windows-Sysmon%4Operational.evtx' --csv
"C:\\Users\\johndoe\\Desktop\\forensic_data\\event_logs\\csv_timeline" --csvf
sysmon_event_log.csv
```

```
Event log details
Flags: None
Chunk count: 28
Stored/Calculated CRC: 3EF9F1C/3EF9F1C
Earliest timestamp: 2023-09-07 08:23:18.4430130
Latest timestamp: 2023-09-07 08:33:00.0069805
Total event log records found: 1,920

Records included: 1,920 Errors: 0 Events dropped: 0

Metrics (including dropped events)
Event ID      Count
1             95
2             76
3             346
4              1
8              44
10             6
11            321
12            674
13            356
16              1

Processed 1 file in 7.1794 seconds
```

-> We look for the keyword scheduler in timeline explorer:

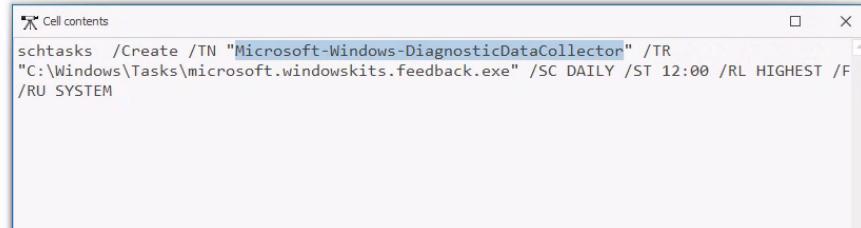
Line	Tag	Record Number	Event Record Id	Time Created	Event Id	Level	Provider	Channel	Process Id	Computer	User Id
1795		1795		2023-09-07 08...	1	Info	Microsoft-Windows-Sysmon	Microsoft-Windows-Sysmon...	5284	HTBVM01	S-1-5-1
1796		1796		2023-09-07 08...	1	Info	Microsoft-Windows-Sysmon	Microsoft-Windows-Sysmon...	5284	HTBVM01	S-1-5-1

-> We only have 2 events, this is very promising.

-> We see from the executable info that the second scheduled tasks matches what we want and that it is created an task name Microsoft-Windows-DiagnositcDataCollctor and

its running the suspicious software we previously identified.

Drag a column header here to group by that column			
	Executable Info	Source File	
▼	s\SYSTEM32\cmd.exe /c install... schtasks /Create /TN "Microsoft-Windows-UpdateTask" /TR "C:\Windows\Tasks\update..."	C:\Users\johndoe\l...	
▶	s\SYSTEM32\cmd.exe /c install... schtasks /Create /TN "Microsoft-Windows-DiagnosticDataCollector" /TR "C:\Windows\...\...	C:\Users\johndoe\l...	



-> Alternatively, we can search for schtasks, as from the microsoft documentation it mentioned that that is how scheduled task are created, through schtasks /create:

The screenshot shows the Timeline Explorer interface. The top menu bar includes File, Tools, Tabs, View, Help. The tabs are set to MFT-J.csv, MFT_backup.csv, and sysmon_event_log.csv. A search bar at the top right contains the term 'schtask'. The main pane displays a list of log entries from the MFT_backup.csv file, with a search results overlay showing multiple occurrences of the term 'schtask' across various log entries.

-> And we would get similar results

```

Timeline Explorer v2.0.0.1
File Tools Tabs View Help
MFT_J.csv MFT_backup.csv symon_event_log.csv
Drag a column header here to group by that column
Source File Payload
ks\update... C:\Users\johndoe\Desktop\forensic_data\kape_output\0\Windows\System32\winevt\logs\Microsoft-Wi... {"EventData": [{"@Name": "RuleName"}, {"@Name": "UtcTime"}, "#text": "2023-09-10T10:00:00Z"}]
> :\\Windows\... C:\Users\johndoe\Desktop\forensic_data\kape_output\0\Windows\System32\winevt\logs\Microsoft-Wi... {"EventData": [{"@Name": "RuleName"}, {"@Name": "UtcTime"}, "#text": "2023-09-10T10:00:00Z"}]

```


Executable Info	Source File
schtasks /Create /TN "Microsoft-Windows-UpdateTask" /TR "C:\Windows\Tasks\update..."	C:\Users\johndoe\Desktop\forensic_data\kape_output\0\Windows\System32\winevt\logs\Microsoft-Wi...
schtasks /Create /TN "Microsoft-Windows-DiagnosticDataCollector" /TR "C:\Windows\..."	C:\Users\johndoe\Desktop\forensic_data\kape_output\0\Windows\System32\winevt\logs\Microsoft-Wi...


```

Cell contents
schtasks /Create /TN "Microsoft-Windows-DiagnosticDataCollector" /TR "C:\Windows\Tasks\microsoft.windowskits.feedback.exe" /SC DAILY /ST 12:00 /RL HIGHEST /F /RU SYSTEM

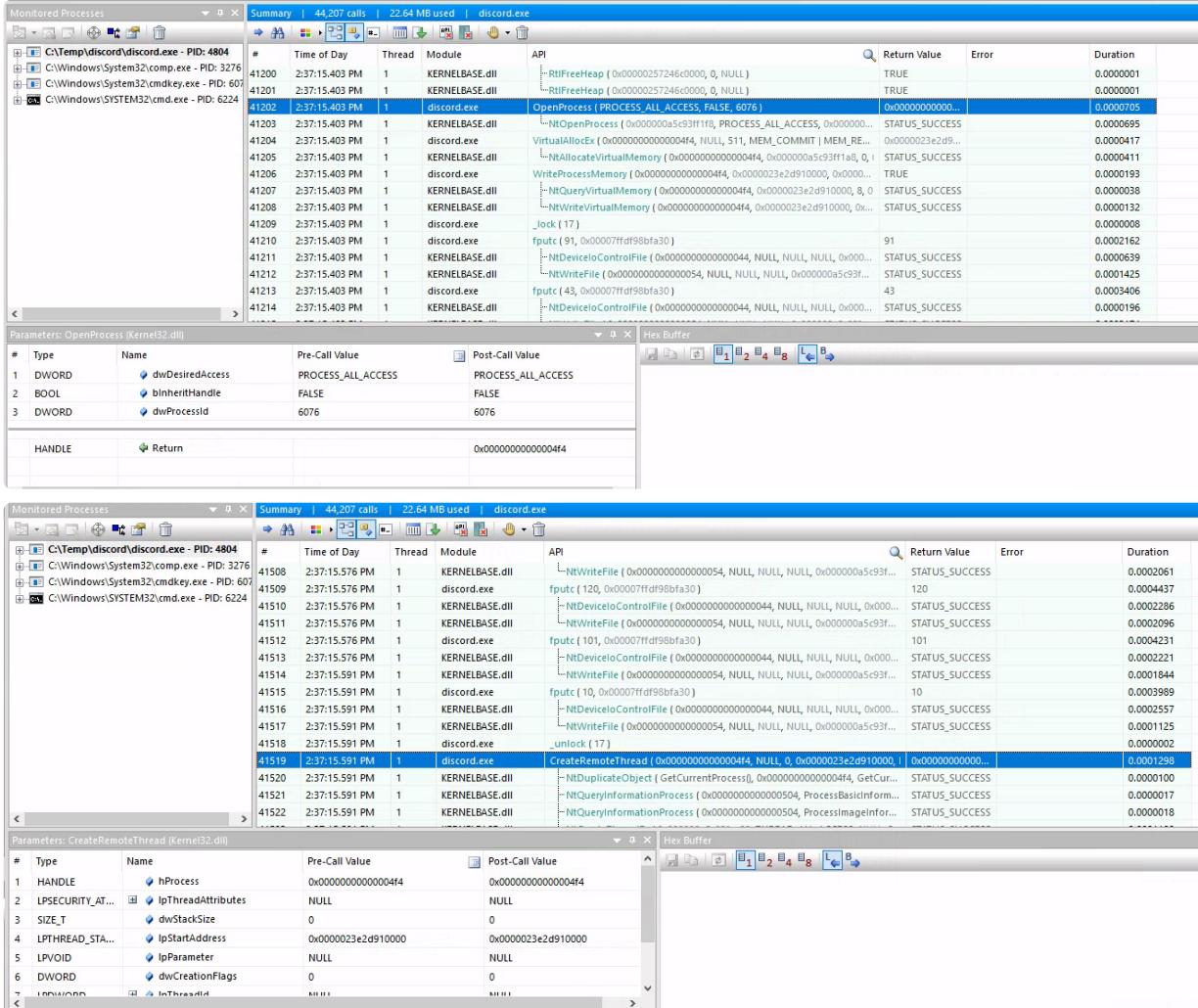
```

- Examine the contents of the file located at "C:\Users\johndoe\Desktop\forensic_data\APMX64\discord.apmx64" using API Monitor. "discord.exe" performed process injection against another process as well. Identify its name and enter it as your answer.
- > We know from the section that first process injection is performed on comp.exe achieved by discord.exe.
- > Looking down, we see another Create Process with a similar Create_Suspended dwCreationFlags.

#	Type	Name	Pre-Call Value	Post-Call Value
1	LPCSTR	lpApplicationName	NULL	NULL
2	LPSTR	lpCommandLine	0x00007ff7fa18adb "C:\Windows\..."	0x00007ff7fa18adb "C:\Windows\..."
3	LPSECURITY_ATTRIBUTES	lpProcessAttributes	NULL	NULL
4	LPSECURITY_ATTRIBUTES	lpThreadAttributes	NULL	NULL
5	BOOL	bInheritHandles	FALSE	FALSE
6	DWORD	dwCreationFlags	CREATE_SUSPENDED	CREATE_SUSPENDED
7	DWORD	InEnvironment	0x00000000	0x00000000

-> Searching for another OpenProcess, we see it opening process cmdkey.exe (with PID of 6076), virtual alloc, writing memory functions and create remote thread (on the

manual memory allocated to cmdkey.exe) happening immediately afterwards, which is indicative of process injection.



-> Hence, we can say that process injection is happening on the process cmdkey.exe.

Practical digital forensic scenario

Question

- Extract and scrutinize the memory content of the suspicious PowerShell process which corresponds to PID 6744. Determine which tool from the PowerSploit repository (accessible at <https://github.com/PowerShellMafia/PowerSploit>) has been utilized within the process, and enter its name as your answer.
-> We performed a keyword search on powersploit in Autopsy and we see it seems to be PowerView on the Keyword Preview.

The screenshot shows the Autopsy 4.2.0 interface. The left sidebar contains various data sources like File Views, Deleted Files, MB File Size, Data Artifacts (including Chromium Extensions, Installed Programs, and OS Information), and Analysis Results (Encryption Suspected, EXIF Metadata, Extension Mismatch Detected, Keyword Hits, User Content Suspected, Web Categories, OS Accounts, Tags, and Reports). The main pane displays a table of results for a keyword search. The table has columns for Name, Keyword Preview, Modified Time, and Location. Two results are listed: 'pagefile.sys' and 'Microsoft-Windows-PowerShell%4Operational.evtx'. The 'Microsoft-Windows-PowerShell%4Operational.evtx' file is selected, and its details are shown in the bottom pane, including its content.

-> We verify this by going to the link given in github and see that PowerView and PowerUp appears, so it seems to be powerview.

The screenshot shows the GitHub page for PowerSploit v3.0.0. It includes a 'Compare' button, a release note from PowerShellMafia dated Dec 19, 2015, stating 275 commits to master since the release, and version information v3.0.0 - o- 9e771d1. Below this, there are sections for 'Features added:', 'Enhancements:', and a detailed list of changes.

- Features added:**
 - PowerView and PowerUp!!! Moving forward this will be the home of these projects. Thank you [@HarmJ0y](#) for all the work that went in to integration and test writing!
 - Pester tests for PowerUp, PowerView, and the CodeExecution module. Full test coverage is desired but cannot be done in the interest of time, at the moment. Moving forward, all new code must be accompanied with Pester tests.
 - PowerSploit includes a .sln now for those who opt to develop PowerSploit in Visual Studio with the PowerShell Tools extension.
- Enhancements:**
 - Invoke-Mimikatz: It now uses the latest build of mimikatz 2.0 alpha (as of 12/14/2015)
 - Everything was normalized to ASCII for a consistent weaponization experience. A Pester test was written to ensure consistent, module-wide ASCII encoding.
 - I removed all versioning comments from functions. Versioning is to be maintained at the module level now.
 - Get-Keystrokes: Added a -PollingInterval parameter

-> Alternative way: Dumping memory and using strings (example in Memory forensics using linux) function to analyse it

-> We can also dump the memory of pid 6744 and perform an string analysis of it.

```
python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.memmap --pid 6744 --dump
```

0xf6fffffc3000	0x200000	0x1000	0x18d5c000	pid.6744.dmp
0xf6fffffc4000	0x200000	0x1000	0x18d5d000	pid.6744.dmp
0xf6fffffc5000	0x200000	0x1000	0x18d5e000	pid.6744.dmp
0xf6fffffc6000	0x200000	0x1000	0x18d5f000	pid.6744.dmp
0xf6fffffc7000	0x200000	0x1000	0x18d60000	pid.6744.dmp
0xf6fffffc8000	0x200000	0x1000	0x18d61000	pid.6744.dmp
0xf6fffffc9000	0x200000	0x1000	0x18d62000	pid.6744.dmp
0xf6fffffc a000	0x200000	0x1000	0x18d63000	pid.6744.dmp
0xf6fffffc b000	0x200000	0x1000	0x18d64000	pid.6744.dmp
0xf6fffffc c000	0x200000	0x1000	0x18d65000	pid.6744.dmp
0xf6fffffc d000	0x200000	0x1000	0x18d66000	pid.6744.dmp
0xf6fffffc e000	0x200000	0x1000	0x18d67000	pid.6744.dmp
0xf6fffffc f000	0x200000	0x1000	0x18d68000	pid.6744.dmp
0xf6fffffd0000	0x200000	0x1000	0x18d69000	pid.6744.dmp
0xf6fffffd1000	0x200000	0x1000	0x18d6a000	pid.6744.dmp
0xf6fffffd2000	0x200000	0x1000	0x18d6b000	pid.6744.dmp
0xf6fffffd3000	0x200000	0x1000	0x18d6c000	pid.6744.dmp
0xf6fffffd4000	0x200000	0x1000	0x18d6d000	pid.6744.dmp
0xf6fffffd5000	0x200000	0x1000	0x18d6e000	pid.6744.dmp
0xf6fffffd6000	0x200000	0x1000	0x18d6f000	pid.6744.dmp
0xf6fffffd7000	0x200000	0x1000	0x18d70000	pid.6744.dmp
0xf6fffffd8000	0x200000	0x1000	0x18d71000	pid.6744.dmp
0xf6fffffd9000	0x200000	0x1000	0x18d72000	pid.6744.dmp
0xf6fffffd a000	0x200000	0x1000	0x18d73000	pid.6744.dmp
0xf6fffffd b000	0x200000	0x1000	0x18d74000	pid.6744.dmp
0xf6fffffd c000	0x200000	0x1000	0x18d75000	pid.6744.dmp
0xf6fffffd d000	0x200000	0x1000	0x18d76000	pid.6744.dmp
0xf6fffffd e000	0x200000	0x1000	0x18d77000	pid.6744.dmp
0xf6fffffd f000	0x200000	0x1000	0x18d78000	pid.6744.dmp
0xf6fffffe0000	0x200000	0x1000	0x18d79000	pid.6744.dmp

-> To use strings function in Windows, we need to obtain system internals as follows to our attack machine on the website: <https://learn.microsoft.com/en-us/sysinternals/downloads/>

```
wget https://download.sysinternals.com/files/SysinternalsSuite.zip
```

-> We then open a web server and transfer it over as follows

```
python -m http.server
```

```
wget http://10.10.16.5:8000/strings64.exe -OutFile strings64.exe
```

```
└── [★]$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.129.228.173 - - [13/Aug/2024 16:52:47] "GET /strings64.exe HTTP/1.1" 200 -
PS C:\Users\johndoe\Desktop> wget http://10.10.16.5:8000/strings64.exe -OutFile strings64.exe
```

-> We then apply string functions to it and search for powerview

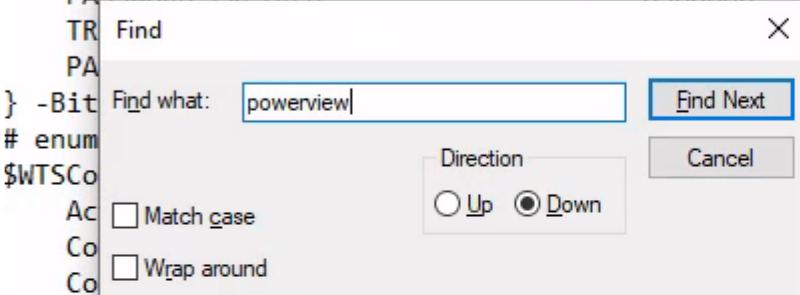
```
.\strings64.exe pid.6744.dmp > powershell_strings.txt
```

```
PS C:\Users\johndoe\Desktop> .\strings64.exe pid.6744.dmp > powershell_strings.txt
```

powershell_strings.txt - Notepad

File Edit Format View Help

MNS_LOGON_ACCOUNT	=	131072
SMARTCARD_REQUIRED	=	262144
TRUSTED_FOR_DELEGATION	=	524288
NOT_DELEGATED	=	1048576
USE_DES_KEY_ONLY	=	2097152
DONT_REQ_PREAUTH	=	4194304
PASSWORD_EXPTRED	=	8388608



Shadow	=	3
Disconnected	=	4
Idle	=	5
Listen	=	6
Reset	=	7
Down	=	8
Init	=	9

```
# the WTSEnumerateSessionsEx result structure
$WTS_SESSION_INFO_1 = struct $Mod PowerView.RDPSessionInfo @{
    ExecEnvId = field 0 UInt32
    State = field 1 $WTSConnectState
    SessionId = field 2 UInt32
    pSessionName = field 3 String -MarshalAs @('LPWStr')
    pHostName = field 4 String -MarshalAs @('LPWStr')
    pUserName = field 5 String -MarshalAs @('LPWStr')
```

-> Hence, we confirmed using 2 different techniques that it is indeed Powerview being utilised in the powershell process.

- Investigate the USN Journal located at "C:\Users\johndoe\Desktop\kapefiles\ntfs\%5C%5C.%5CC%3A\$Extend\$UsnJrnI%3A\$J" to determine how "advancedip_scanner.exe" was introduced to the compromised system. Enter the name of the associated process as your answer. Answer format: .exe
- We first parse the usn journal into CSV format as shown below and open it in time explorer.

```
.\\MFTECmd.exe -f
'C:\\Users\\johndoe\\Desktop\\kapecfiles\\ntfs\\%5C%5C.%5CC%3A\\$Extend\\$UsnJrn\\
%3A$J' --csv C:\\Users\\johndoe\\Desktop --csvf usn_j.csv
```

- We filter for the name advanced_ip scanner

Line	Tag	Update Timestamp	Parent Path	Name	Extension	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	Update Se
220564		2023-08-10 09:20:04		advanced.zip	.zip	112356	2			
220565		2023-08-10 09:20:04		advanced.zip	.zip	112356	2			
220566		2023-08-10 09:20:06		advanced.zip	.zip	112356	2			
220567		2023-08-10 09:20:06		advanced.zip	.zip	112356	2			
220568		2023-08-10 09:20:08		advanced.zip	.zip	112356	2			
220569		2023-08-10 09:20:08		advanced.zip	.zip	112356	2			
220570		2023-08-10 09:20:10		advanced.zip	.zip	112356	2			
220571		2023-08-10 09:20:10		advanced.zip	.zip	112356	2			
220572		2023-08-10 09:20:11		advanced.zip	.zip	112356	2			
220573		2023-08-10 09:20:11		advanced.zip	.zip	112356	2			
220574		2023-08-10 09:20:26		Advanced IP Scann...		112357	2			
220575		2023-08-10 09:20:26		Advanced IP Scann...		112357	2			
220576		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220577		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220578		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220579		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220580		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220581		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220582		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220583		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220584		2023-08-10 09:20:26		advanced_ip_scann...	.qm	112360	2			
220585		2023-08-10 09:20:26		advanced_ip_scann...	.qm	112360	2			

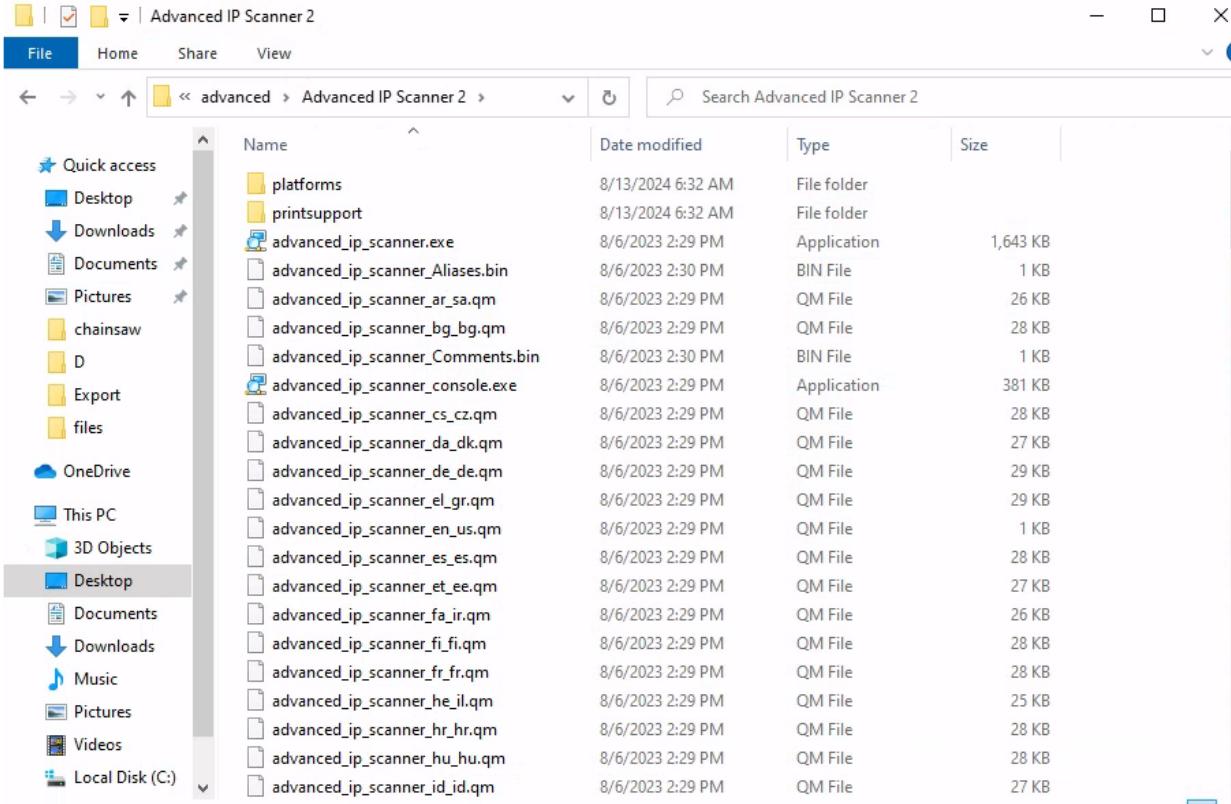
- We also see that right before advanced_ip_scanner.exe there is an zip file called advanced.zip.

Line	Tag	Update Timestamp	Parent Path	Name	Extension	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence Number	Par
220564		2023-08-10 09:20:04		advanced.zip	.zip	112356	2			
220565		2023-08-10 09:20:04		advanced.zip	.zip	112356	2			
220566		2023-08-10 09:20:06		advanced.zip	.zip	112356	2			
220567		2023-08-10 09:20:06		advanced.zip	.zip	112356	2			
220568		2023-08-10 09:20:08		advanced.zip	.zip	112356	2			
220569		2023-08-10 09:20:08		advanced.zip	.zip	112356	2			
220570		2023-08-10 09:20:10		advanced.zip	.zip	112356	2			
220571		2023-08-10 09:20:10		advanced.zip	.zip	112356	2			
220572		2023-08-10 09:20:11		advanced.zip	.zip	112356	2			
220573		2023-08-10 09:20:11		advanced.zip	.zip	112356	2			
220574		2023-08-10 09:20:26		Advanced IP Scann...		112357	2			
220575		2023-08-10 09:20:26		Advanced IP Scann...		112357	2			
220576		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220577		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220578		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220579		2023-08-10 09:20:26		advanced_ip_scann...	.exe	112358	2			
220580		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220581		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220582		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220583		2023-08-10 09:20:26		advanced_ip_scann...	.bin	112359	2			
220584		2023-08-10 09:20:26		advanced_ip_scann...	.qm	112360	2			
220585		2023-08-10 09:20:26		advanced_ip_scann...	.qm	112360	2			

-> Hence, we search for this keyword in autopsy and decide to extract the file to see whether it contains advanced_ip_scanner.exe

Name	Keyword Preview	Modified Time	Location
advanced.zip	slac	2023-08-10 09:20:11 UTC	/img_fulldisk.raw.001/Users/john
advanced.zip-s	0<<advanced.zip<>19,\${' Ifile0adv	2023-08-10 09:20:11 UTC	/img_fulldisk.raw.001/Users/john
\$MFT	xf <<advanced.zip<>c:\windows\system32	2023-08-10 02:16:05 UTC	/img_fulldisk.raw.001/\$MFT
Security.evtx	<<advanced.zip<>+,<sess4fc295ee>	2023-08-10 09:44:10 UTC	/img_fulldisk.raw.001/Windows/E
pagefile.sys	,pf <<advanced.zip<><advanced.zip<>adva	2023-08-10 00:22:55 UTC	/img_fulldisk.raw.001/pagefile.sy
\$UsnJrnls:\$	2023-08-10 01:18:48 UTC	/img_fulldisk.raw.001/\$Extend/\$U	

-> Unzipping the file we confirmed our suspicion and saw advanced_ip_scanner.exe in it.



-> As such, we only have to look at the windows event log for file creation of this advanced.zip file and find the corresponding process associated with it.

-> We go back to Autopsy and examine the sysmon log file and saw that it indeed

recorded the event for file creation and the associated process (image) is rundll32.exe.

The screenshot shows the WinHex application interface. At the top, there are three tabs: 'Keyword search 1 - powersploit', 'Keyword search 2 - advanced_ip_sc...', and 'Keyword search 3 - advanced.zip'. Below these tabs, the status bar indicates '17 Result'. The main window displays a table of search results with columns: Name, Keyword Preview, Modified Time, and Location. One row in the table is highlighted, corresponding to the file 'Microsoft-Windows-Sysmon%4Operational.evtx'. Below the table, there are tabs for Hex, Text, Application, File Metadata, OS Account, Data Artifacts, Analysis Results, Context, Annotations, and Other Occurrences. The 'Text' tab is selected, showing the content of the file. The content includes several lines of log data, such as:

```
[microsoft-windows-sysmon%4operational.evtx, gF8#Y
RuleName
UtcTime
ProcessGuid
    ProcessId
Image
TargetFilename
CreationUtcTime
User
2023-08-10 09:17:28,454
C:\Windows\System32\rundll32.exe
C:\Users\johndoe\AppData\Local\advanced.zip
2023-08-10 09:17:28,454
DESKTOP-VQJOLVH\johndoe
t.exe
Microsoft-Windows-Sysmon_8pW*
Microsoft-Windows-Sysmon/Operational
```

Skills Assessment

Scenario:

- Upon identifying signs of data exfiltration from an unusual process on a system, the SOC manager tasked you with conducting a forensic investigation through **Velociraptor**.
- Once you've established a connection to the target of this section via RDP, visit the URL <https://127.0.0.1:8889/app/index.html#/search/all> and log in using the credentials: **admin/password**. After logging in, click on the circular symbol adjacent to **Client ID**. Subsequently, select the displayed **Client ID** and click on **Collected**.
- Answer the questions below through Velociraptor collections that gather artifacts similar to the ones presented in this module.
- Note:** You can initiate Velociraptor collections in the same manner as Velociraptor hunts.

Questions

- Using VAD analysis, pinpoint the suspicious process and enter its name as your

answer. Answer format: __.exe

-> We first try different collection methods that uses VAD on Velociraptor and the artifact that proved to work is Exchange.Windows.Detection.Malfind, where it looks for memory regions that has read, write and execute and uses yara rules to look for suspicious process as shown below:

State	Hunt ID	Description	Created	Started	Expires	Scheduled	Created
☒	H.CQU1CJVV5T0J4	vad_test2	2024-08-14T02:28:31Z	2024-08-14T02:28:31Z	2024-08-21T02:25:44Z	1	admin
☒	H.CQU13FF9H03KM	vad_susp	2024-08-14T02:09:01Z	2024-08-14T02:09:01Z	2024-08-21T02:07:50Z	1	admin
Artifact Names				Total scheduled	1		
Hunt ID				Finished clients	1		
Creator				Download Results			
Creation Time				Available Downloads			
Expiry Time				H.CQU1CJVV5T0J4			
State				Uncompressed	95 Kb		
Ops/Sec				Compressed	76 Kb		
CPU Limit				Container	12		
IOPS Limit				Files			
Parameters				Started	2024-08-14T02:29:17Z		
Exchange.Windows.Detection.Malfind				Duration (Sec)	1		
				SHA256 Hash	3f6f316d7ac36d9e5c194c0545627ba4484216fee948b1fdf6efaf359d391ea		

-> We download the result and open the csv file, we immediately found the suspicious process reverse.exe, which seems to be identified with the rule shellcode_stack_strings and indicates shell code execution.

-> The name is also suspicious as well, which hints at a reverse shell.

-> We can also look at it in velociraptor notebook section and see that it is likely an cobalt strike payload.

State	Hunt ID	Description	Created	Started	Expires	Scheduled	Created
X	H.CQU6A9CVUMUEE	network	2024-08-14T08:04:53Z	2024-08-14T08:04:53Z	2024-08-21T08:02:54Z	1	admin
■	H.CJ0H7UEPS5CUK	Users	2023-08-31T22:28:09Z	2023-08-31T22:28:09Z	2023-09-07T22:27:00Z	102	admin

-> This hints that the next thing we should look at is cobalt strike related artifact analysis if we were to do analysis on C2 server that it connects back or just further analysis of the payload.

-> It also has pid of 3988.

- Determine the IP address of the C2 (Command and Control) server and enter it as your answer.
- > From the previous question, we use the windows.carving.cobalt strike, filtering with regex of process name of reverse, since we know it is an cobalt strike payload from the previous question and we can extract the configuration.

->We get the following below:

State	Hunt ID	Description	Created	Started	Expires	Scheduled	Created
X	H.CQU6E47G1MLES	cobalt 3	2024-08-14T08:13:04Z	2024-08-14T08:13:04Z	2024-08-21T08:12:34Z	1	admin
X	H.CQU6DEQUN37V6	cobalt strike 2	2024-08-14T08:11:39Z	2024-08-14T08:11:39Z	2024-08-21T08:11:19Z	1	admin
X	H.CQU6CPFU9N012	cobalt-strike	2024-08-14T08:10:13Z	2024-08-14T08:10:13Z	2024-08-21T08:09:19Z	1	admin
X	H.CQU6A9CVUMUEE	network	2024-08-14T08:04:53Z	2024-08-14T08:04:53Z	2024-08-21T08:02:54Z	1	admin

State	Hunt ID	Description	Created	Started	Expires	Scheduled	Creator
X	H.CQU6E47G1MLES	cobalt 3	2024-08-14T08:13:04Z	2024-08-14T08:13:04Z	2024-08-21T08:12:34Z	1	admin
X	H.CQU6DEQUN37V6	cobalt striek 2	2024-08-14T08:11:39Z	2024-08-14T08:11:39Z	2024-08-21T08:11:19Z	1	admin
X	H.CQU6CPFU9N812	cobalt-strike	2024-08-14T08:10:13Z	2024-08-14T08:10:13Z	2024-08-21T08:09:19Z	1	admin
X	H.CQU6A9CVUMUEE	network	2024-08-14T08:04:53Z	2024-08-14T08:04:53Z	2024-08-21T08:02:54Z	1	admin

```

"SleepTime" : 60000
"MaxGetSize" : 1048576
"PublicKey" :
"3B 81 9F 30 0d 00 09 2a 86 48 86 f7 0d 01 01 01 05 00 03 81 8d 00 30
81 89 02 81 81 00 a7 38 cd e7 5...""
"C2Server" : "3.19.219.4:/ptj"
"UserAgent" :
"Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64;
Trident/6.0)"
"PostURI" : "/submit.php"
"HttpGetHeader" : "Cookie"
"HttpPostHeader" : "Content-Type: application/octet-streamid"
"DNSIdle" : "0.0.0.0"
"GetVerb" : "GET"
"PostVerb" : "POST"
"SpawnTox86" : "%windir%\syswow64\rundll32.exe"
"SpawnTox64" : "%windir%\sysnative\rundll32.exe"
"ProxyType" : "IE settings"

```

-> And we see that the C2 server has an ip address of 3.19.219.4

- Determine the registry key used for persistence and enter it as your answer.
- > We look at the artifact that contains collects various persistence mechanism in windows (including event filers, startup items..).

Create Hunt: Select artifacts to collect

- `Exchange.Linux.Detection.SSHKeyFileCmd`
- `Exchange.MacOS.System.Man`
- `Exchange.Windows.Analysis.SuspiciousWMICor`
- `Exchange.Windows.Detection.RemoteIconForce`
- `Exchange.Windows.Persistence.VscodeTasks`
- `Exchange.Windows.PersistenceSniper`
- `Exchange.Windows.Registry.NetshHelperDLLs`
- `Windows.EventLogs.Kerbroasting`
- `Windows.EventLogs.ScheduledTasks`
- `Windows.Packs.Persistence`
- `Windows.Persistence.Debug`

Windows.Packs.Persistence

Type: client

This artifact pack collects various persistence mechanisms in Windows.

Source WMI Event Filters

```
1 SELECT * FROM Artifact.Windows.Persistence.PermanentWMIEvents()
2
```

Source Startup Items

```
1 SELECT * FROM Artifact.Windows.Sys.StartupItems()
2
```

Source Debug Bootstrapping

[Configure Hunt](#) [Select Artifacts](#) [Configure Parameters](#) [Specify Resources](#) [Review](#) [Launch](#)

-> We saw the suspicious reverse process that we previously identified!

Windows.Packs.Persistence/Startup Items

Name	OSPath	Details	Enabled	Upload	FlowId	ClientId	Fqdn
OneDriveS	HKEY_USERS\S-1-5-20\Software\Microsoft\Windows\CurrentVersion\Run\OneDriveSetup	C:\Windows\SysWOW64\OneDriveSetup.exe /thfirstsetup	disabled		F.CQU7C13B	C.e096772397	E-IMSM.H 9e1134 CORP
OneDriveS	HKEY_USERS\S-1-5-19\Software\Microsoft\Windows\CurrentVersion\Run\OneDriveSetup	C:\Windows\SysWOW64\OneDriveSetup.exe /thfirstsetup	disabled		F.CQU7C13B	C.e096772397	E-IMSM.H 9e1134 CORP
SecurityH	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\SecurityHealth	%windir%\system32\SecurityHealthSystray.exe	disabled		F.CQU7C13B	C.e096772397	E-IMSM.H 9e1134 CORP
VMware User Process	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\VMware User Process	"C:\Program Files\VMware Tools\vmtoolsd.exe" -n vmusr	disabled		F.CQU7C13B	C.e096772397	E-IMSM.H 9e1134 CORP
reverse	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\reverse	C:\Users\joseph\AppData\Local\reverse.exe	disabled		F.CQU7C13B	C.e096772397	E-IMSM.H 9e1134 CORP

-> We see the suspicious reverse shell under the key

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run which we previously identified!

- Determine the folder that contains all Mimikatz-related files and enter the full path as your answer.
 - > We search for the yara.search artifact (for file look up) with mimikatz as initial guess/hunt.
 - > The motivation behind is because mimkatz is an malware that can be detected through Yara signature-based detection, so yara is an solid choice here.

Create Hunt: Configure artifact parameters

The screenshot shows the 'Configure Parameters' tab of a Yara search configuration window. It includes fields for 'nameRegex' (containing '(exe|txt|dll|php)\$'), 'yaraRule' (containing a Yara rule for 'mimikatz'), and 'NTFS_CACHE_TIME' (set to 1000000). Below the tabs are 'Configure Hunt', 'Select Artifacts', 'Configure Parameters' (highlighted in green), 'Specify Resources', 'Review', and 'Launch' buttons.

-> And we get the result of the directory

Rule	HitOffset	HitContext	FileName	Size	ModTime	Upload	FlowId	ClientId	Fqdn
Hit	1461	mimikatz	\\.\C:\Users\Administrator\PowerShell_transcript\20230906\PowerShell_transcript.E-CORP.fkw3Ddk.20230906204324.txt	29367	2023-09-06T18:43:45Z	F.CQUM9JDJU1	C.e0967723979	E-FLK.H	c1134 CORP
Hit	714252	m i m i k a t \\.\C:\Users\j0seph\AppData\Local\mimik\Win32\mimikatz.exe	z	10844	2022-09-16 19T14:43:56Z	F.CQUM9JDJU1	C.e0967723979	E-FLK.H	c1134 CORP
Hit	20133	mimikatz	\\.\C:\Users\j0seph\AppData\Local\mimik\Win32\mimilib.dll	31744	2022-09-19T14:43:28Z	F.CQUM9JDJU1	C.e0967723979	E-FLK.H	c1134 CORP
Hit	13414	m i m i k a t \\.\C:\Users\j0seph\AppData\Local\mimik\Win32\mimilove.exe	z	25088	2022-09-19T14:43:18Z	F.CQUM9JDJU1	C.e0967723979	E-FLK.H	c1134 CORP
Hit	7982	m i m i k a t \\.\C:\Users\j0seph\AppData\Local\mimik\Win32\mimispool.dll	z	10240	2022-09-19T14:43:16Z	F.CQUM9JDJU1	C.e0967723979	E-FLK.H	c1134 CORP
Hit	935800	m i m i k a t \\.\C:\Users\j0seph\AppData\Local\mimik\x64\mimikatz.exe	z	13552	2022-09-19T14:44:40Z	F.CQUM9JDJU1	C.e0967723979	E-FLK.H	c1134 CORP

C:\Users\j0seph\AppData\Local\mimik\

-> However the above is not the most reliable method (can be easily evaded through tweaking file names), we could also search for detecting mimikatz using a more reliable rule (based on the property of the mimikatz) which we searched below:

mimikatz detection rule yara windows

All Images Videos News Maps Shopping Chat Settings

Always private Australia Safe search: moderate Any time

https://adsecurity.org › ?p=1567

Detecting Mimikatz Use - Active Directory Security

YARA is multi-platform, running on Windows, Linux and Mac OS X, and can be used through its command-line interface or from your own Python scripts with the yara-python extension. Based on the published data, this data enables detection of the Mimikatz exe, dll, and artifacts such as kirbi ticket...

...
YARA is multi-platform, running on Windows, Linux and Mac OS X, and can be used through its command-line interface or from your own Python scripts with the yara-python extension.

Based on the published data, this data enables detection of the Mimikatz exe, dll, and artifacts such as kirbi ticket files, as well as WCE.
Here's the data:

```
/* Benjamin DELPY `gentilkiwi`
http://blog.gentilkiwi.com
benjamin@gentilkiwi.com
Licence : http://creativecommons.org/licenses/by/3.0/fr/
*/
rule mimikatz
{
meta:
description = "mimikatz"
author = "Benjamin DELPY (gentilkiwi)"
tool_author = "Benjamin DELPY (gentilkiwi)"
strings:
$exe_x86_1 = { 89 71 04 89 [0-3] 30 8d 04 bd }
$exe_x86_2 = { 89 79 04 89 [0-3] 38 8d 04 b5 }

$exe_x64_1 = { 4c 03 d8 49 [0-3] 8b 03 48 89 }
$exe_x64_2 = { 4c 8b df 49 [0-3] c1 e3 04 48 [0-3] 8b cb 4c 03 [0-3] d8 }
```

mimikatz / kiwi_passwords.yar

gentilkiwi [new] mimikatz lsadump::postzerologon, to reinit DC password both in ...

```
Code Blame 104 lines (80 loc) · 2.77 KB

1  /* Benjamin DELPY `gentilkiwi`  
2   https://blog.gentilkiwi.com  
3   benjamin@gentilkiwi.com  
4   Licence : https://creativecommons.org/licenses/by/4.0/  
5 */  
6 rule mimikatz  
7 {  
8     meta:  
9         description      = "mimikatz"  
10        author          = "Benjamin DELPY (gentilkiwi)"  
11        tool_author      = "Benjamin DELPY (gentilkiwi)"  
12  
13     strings:  
14         $exe_x86_1       = { 89 71 04 89 [0-3] 30 8d 04 bd }  
15         $exe_x86_2       = { 8b 4d e? 8b 45 f4 89 75 e? 89 01 85 ff 74 }  
16  
17         $exe_x64_1       = { 33 ff 4? 89 37 4? 8b f3 45 85 c? 74}  
18         $exe_x64_2       = { 4c 8b df 49 [0-3] c1 e3 04 48 [0-3] 8b cb 4c 03 [0-3] d8 }  
19  
20         $dll_1           = { c7 0? 00 00 01 00 [4-14] c7 0? 01 00 00 00 }  
21         $dll_2           = { c7 0? 10 02 00 00 ?? 89 4? }  
22  
23         $sys_x86          = { a0 00 00 00 24 02 00 00 40 00 00 00 [0-4] b8 00 00 00 6c 02 00 00 40 00 00 00 }  
24         $sys_x64          = { 88 01 00 00 3c 04 00 00 40 00 00 00 [0-4] e8 02 00 00 f8 02 00 00 40 00 00 00 }  
25  
26     condition:  
27     (all of ($exe_x86_*)) or (all of ($exe_x64_*)) or (all of ($dll_*)) or (any of ($sys_*))  
28 }  
29
```

https://github.com/gentilkiwi/mimikatz/blob/master/kiwi_passwords.yar

-> Where we use the following rule for searching mimikatz

```
rule mimikatz  
{  
    meta:  
        description      = "mimikatz"  
        author          = "Benjamin DELPY (gentilkiwi)"  
        tool_author      = "Benjamin DELPY (gentilkiwi)"  
  
    strings:  
        $exe_x86_1       = { 89 71 04 89 [0-3] 30 8d 04  
bd }  
        $exe_x86_2       = { 8b 4d e? 8b 45 f4 89 75 e?  
89 01 85 ff 74 }  
  
        $exe_x64_1       = { 33 ff 4? 89 37 4? 8b f3 45  
85 c? 74}  
        $exe_x64_2       = { 4c 8b df 49 [0-3] c1 e3 04  
48 [0-3] 8b cb 4c 03 [0-3] d8 }  
  
        $dll_1           = { c7 0? 00 00 01 00 [4-14] c7
```

```

0? 01 00 00 00 }
$dll_2 = { c7 0? 10 02 00 00 ?? 89 4? }

$sys_x86 = { a0 00 00 00 24 02 00 00 40
00 00 00 [0-4] b8 00 00 00 6c 02 00 00 40 00 00 00 }

$sys_x64 = { 88 01 00 00 3c 04 00 00 40
00 00 00 [0-4] e8 02 00 00 f8 02 00 00 40 00 00 00 }

condition:
(all of ($exe_x86_*)) or (all of ($exe_x64_*)) or (all
of ($dll_*)) or (any of ($sys_*))
}

```

Create Hunt: Configure artifact parameters

nameRegex `(exe|txt|dll|php)$`

AlsoUpload Also upload matching files.

yaraRule

```

rule mimikatz
{
    meta:
        description      = "mimikatz"
        author          = "Benjamin DELPY (gentilkiwi)"
        tool_author     = "Benjamin DELPY (gentilkiwi)"

    strings:
        $exe_x86_1      = { 89 71 04 89 [0-3] 30 8d 04 bd }
        $exe_x86_2      = { 8b 4d e? 8b 45 f4 89 75 e? 89 01 85 ff 74 }

        $exe_x64_1      = { 33 ff 4? 89 37 4? 8b f3 45 85 c? 74}
        $exe_x64_2      = { 4c 8b df 49 [0-3] c1 e3 04 48 [0-3] 8b cb 4c 03
                           [0-3] d8 }

```

Configure Hunt Select Artifacts Configure Parameters Specify Resources Review Launch

-> Hunting with the more accurate yara rule, we see an more accurate detection result.

Windows.Search.Yara

Rule	HitOffset	HitContext	FileName	Size	ModTime	Upload	FlowId	ClientId	Fqdn
mimika	1821796	♦q ♦0♦ ♦	\.\C:\Users\j0seph\AppData\Local\mimik\Win32\mimikatz.exe	108441	2022-09-6 19T14:43:56Z	F.CQUMF0BUDAJQ	C.e0967723979c1	E-U.H 134	CORP
mimika	1890	♦ ♦D\$ ♦	\.\C:\Users\j0seph\AppData\Local\mimik\Win32\mimilib.dll	31744	2022-09-19T14:43:20Z	F.CQUMF0BUDAJQ	C.e0967723979c1	E-U.H 134	CORP
mimika	1278752	3♦A♦7L♦♦E♦♦t	\.\C:\Users\j0seph\AppData\Local\mimik\x64\mimikatz.exe	135526	2022-09-4 19T14:44:40Z	F.CQUMF0BUDAJQ	C.e0967723979c1	E-U.H 134	CORP
mimika	2395	♦ I♦ A♦	\.\C:\Users\j0seph\AppData\Local\mimik\x64\mimilib.dll	37376	2022-09-19T14:44:02Z	F.CQUMF0BUDAJQ	C.e0967723979c1	E-U.H 134	CORP

- Determine the Microsoft Word document that j0seph recently accessed and enter its name as your answer. Answer format: __.DOCX

-> We search for recently used document in Velociraptor and we come across the interesting artifact Windows.Registry.RecentDocs, which extracts RecentDocs MRU (most

recently used) from the target.

Create Hunt: Select artifacts to collect



Exchange.Windows.Triage.Sysonline

- Exchange.Windows.Forensics.RecentFileCache
- Exchange.Windows.Office.MRU
- Exchange.Windows.Triage.Sysonline
- Windows.Applications.SBECmd
- Windows.Forensics.RecentApps
- Windows.Forensics.Timeline
- Windows.Forensics.Usn
- Windows.Registry.RDP
- Windows.Registry.RecentDocs**
- Windows.System.Amcache
- Windows.System.PowerShell.ModuleAnalysisCache

Windows.Registry.RecentDocs

Type: client

Author: Matt Green - @mgreen27

This artifact extracts RecentDocs MRU from the target.

By default the artifact will target all users on the machine when run in live mode but can be targeted directly using the HiveGlob parameter.

Output includes LastWriteTime of key and a list of MRU items in the order specified in the MRUListEx key value. MruEntries has the format: [KeyName] := [Parsed Key value]

Available filters include: - Time bounds to select LastWrite timestamp within time ranges. - EntryRegex to target specific entry values - UserRegex to target specific users. Note: this filter does not work when using HiveGlob. - SidRegex to target a specific SID.

Note: both UserRegex and SidRegex does not work when using HiveGlob and all MRU will be returned.

Parameters

-> We filter for the j0seph user and .DOCX regex as follows in parameter configuration:

Create Hunt: Configure artifact parameters

- Artifact

- Windows.Registry.RecentDocs

Filter artifact parameter name

HiveGlob optional hive glob to target for offline processing.

DateAfter --/--/---- --:-- X UTC

DateBefore --/--/---- --:-- X UTC

EntryRegex .DOCX\$

UserRegex j0seph

SidRegex .

Configure Hunt **Select Artifacts** **Configure Parameters** **Specify Resources** **Review** **Launch**

-> Inspecting the result of the artifact collected, we see that insurance.DOCX is the .DOCX file last accessed.

Windows.Registry.RecentDocs

LastWriteTime	Type	MruEntries	Key	HiveName	Username	UUID	FlowId	ClientId	Fqdn
2023-09-06T18:46:41Z	.DOC	X	\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\MRULISTEx	C:\Users\j0seph	j0seph	S-1-5-21-1019847786-291584978-2158772757-1000	F.CQUNP13T	C.e09677239	E-CORP

10 25 30 50 Showing 1 to 1 of 1

<< 0 >> Goto Page

-> Alternatively, we can also ignore the DOCX\$ in the EntryRegex (while keeping the j0seph for UserRegex) and still obtain the result, but we just have to scroll the results obtained to see insurance.DOCX:

ice"								
2023-09-06T18:46:41Z	.DOCX	["\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\h\NTUSER.DAT", "0 := \.DOCX\MRUListEx", "insurance.DOC", "X"]	C:\Users\j0seph\j0seph	S-1-5-21-1019847786-291584978-	F.CQUNK4K8	C.e09677239	E-0807U.H	79c1134
2023-09-06T18:46:37Z	.EML	["\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\h\NTUSER.DAT", "0 := \.EML\MRUListEx", "resource.EML"]	C:\Users\j0seph\j0seph	S-1-5-21-1019847786-291584978-	F.CQUNK4K8	C.e09677239	E-0807U.H	79c1134
2023-09-03T21:56:27Z	.log	["\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\h\NTUSER.DAT", "0 := \.log\MRUListEx", "\.log := WinSW-x64.err.log"]	C:\Users\j0seph\j0seph	S-1-5-21-1019847786-291584978-	F.CQUNK4K8	C.e09677239	E-0807U.H	79c1134

Summary for skills assessment

- Another way to conduct forensic investigation is to use Velociraptor (or tools like it) using the different artifact collection methods.
 - E.g. We don't always need the memory dump and/or kape artifact in front of us and analyse it with the EricZimmermann tools and more ..., sometimes its not possible or less convenient, though it is another way and can give us more detailed insight.