**Introduction to KAPE:**

Kroll Artifact Parser and Extractor (KAPE) parses and extracts Windows forensics artifacts. It is a tool that can significantly reduce the time needed to respond to an incident by providing forensic artifacts from a live system or a storage device much earlier than the imaging process completes.

KAPE serves two primary purposes, 1) collect files and 2) process the collected files as per the provided options. For achieving these purposes, KAPE uses the concept of targets and modules. Targets can be defined as the forensic artifacts that need to be collected. Modules are programs that process the collected artifacts and extract information from them. We will learn about them in the upcoming tasks.

**How it works**

KAPE is extensible and highly configurable. In essence, the KAPE binary ﻿ collects files and processes them as per the provided configuration.

The collection of files (targets) KAPE adds the files to a queue and copies them in two passes. In the first pass, it copies the files that it can. This works for files that the OS has not locked. The rest of the files are passed to a secondary queue. The secondary queue is processed using a different technique that uses raw disk reads to bypass the OS locks and copy the files. The copied files are saved with original timestamps and metadata and stored in a similar directory structure.

Once the data is collected, KAPE can process it using modules. The modules can be independent binaries that run on the collected data and process them to extract information. For example, KAPE will collect and copy the Prefetch file to our target destination during the target collection. Running a Prefetch Parser (PECmd) module on this target will extract the prefetch file and save it in a CSV file.

A blue square with yellow text

Description automatically generated

As the above image shows, KAPE can extract targets from a Live system, a mounted image, or the [F-response](https://www.f-response.com/) utility. KAPE does not need to be installed. It is portable and can be used from network locations or USB drives.

In KAPE's lexicon, Targets are the artifacts that need to be collected from a system or image and copied to our provided destination. For example, as we learned in the last room, Windows Prefetch is a forensic artifact for evidence of execution so that we can create a Target for it. Similarly, we can also create Targets for the registry hives. In short, Targets copy files from one place to another.

When we open the Targets directory of KAPE, this is what we will see:

A screenshot of a computer

Description automatically generated

The last four files at the bottom are guides and templates to create Targets and Compound Targets of our own. We will discuss Compound Targets later in this task. As you can see, the targets are grouped into different directories. Let's check out the Windows directory to see what we have:

A screenshot of a computer

Description automatically generated

We can see different .tkape extension files. This is how a Target is defined for KAPE. A TKAPE file contains information about the artifact that we want to collect, such as the path, category, and file masks to collect. As an example, below is how the Prefetch Target is defined.

A screenshot of a computer

Description automatically generated

This TKAPE file tells KAPE to collect files with the file mask \*.pf from the path C:\Windows\prefetch and C:\Windows.old\prefetch.

Notice that we have the C:\Windows.old path listed here as well. This path contains files retained after Windows has updated to a new version. For forensic analysis, we can also find interesting historical artifacts from this directory.

## Compound Targets:

﻿KAPE also supports Compound Targets. These are Targets that are compounds of multiple other targets. As mentioned in the previous tasks, KAPE is often used for quick triage collection and analysis. The purpose of KAPE will not be fulfilled if we have to collect each artifact individually. Therefore, Compound Targets help us collect multiple targets by giving a single command. Examples of Compound Targets include !BasicCollection, !SANS\_triage and KAPEtriage. We can view the Compound Targets on the path KAPE\Targets\Compound. The following image shows what a Compound Target for evidence of execution looks like:

A screenshot of a computer

Description automatically generated

The above Compound Target will collect evidence of execution from Prefetch, RecentFileCache, AmCache, and Syscache Targets.

## !Disabled

This directory contains Targets that you want to keep in the KAPE instance, but you don't want them to appear in the active Targets list.

## !Local

If you have created some Targets that you don't want to sync with the KAPE Github repository, you can place them in this directory. These can be Targets that are specific to your environment. Similarly, anything not present in the Github repository when we update KAPE will be moved to the !Local directory.

Modules, in KAPE's lexicon, run specific tools against the provided set of files. Their goal is not to copy files from one place to another but rather run some command and store the output. Generally, the output is in the form of CSV or TXT files.

This is what the Modules directory looks like in KAPE:

A screenshot of a computer program

Description automatically generated

Similar to the previous task, we see guides and templates for creating Modules and Compound Modules. We also see the !Disabled, !Local and Compound directories, which are similar to what we saw in the previous task. We will not discuss these again, as we discussed them in the last task. We see that most of the Modules are grouped together in different directories. One thing we find different is the bin directory.

## The bin directory:

The bin directory contains executables that we want to run on the system but are not natively present on most systems. KAPE will run executables either from the bin directory or the complete path. An example of files to be kept in the bin directory are Eric Zimmerman's tools, which are generally not present on a Windows system.

A screenshot of a computer

Description automatically generated

Here we see files with the .mkape extension. These are understood as Modules by KAPE. Let's open an MKAPE file and see how it is structured. The following image shows the Windows\_IPConfig MKAPE file.

A screenshot of a computer

Description automatically generated

Notice that the MKAPE file tells KAPE about the executable that has to be run, the command line parameters of the executable file, the output export format, and the filename to export to. But what if the executable that we want to run is not present on the system? This brings us to the bin directory.

A screenshot of a computer

Description automatically generated

Now that we have learned about the different components of KAPE let's take it for a test drive. Double-click to open the gkape.exe file. You will see the following Window:

A screenshot of a computer

Description automatically generated

Here you can see that there are different options, but most are disabled. To collect Targets We will go ahead by enabling the Use Target Options checkbox. This will enable the options present in the left half of the Window:

A screenshot of a computer

Description automatically generated

If we want to perform forensics on the same machine on which KAPE is running, we will provide C:\ for the Target source. We can select the target destination of our choice. All the triage files will be copied to the Target destination that we provide.

A screenshot of a computer

Description automatically generated

Here, the Flush checkbox will delete all the contents of the Target destination, so we have to be careful when using that. We have disabled the Flush checkbox so that it does not delete data already present in the directories. Add %d will append date info to the directory name where the collected data is saved. Similarly, Add %m will append machine info to the Target destination directory. We can select our desired Target from the list shown above. The Search bar helps us search for the names of the desired Targets quickly.

A screenshot of a computer

Description automatically generated

We can select if we want to process Volume Shadow Copies by enabling Process VSCs. We can select the transfer checkbox if we want to transfer the collected artifacts through an SFTP server or an S3 bucket. For transfer, the files must be enclosed in a container, which can be Zip, VHD, or VHDX. Similarly, we can provide exclusions based on SHA-1, and KAPE will not copy the excluded files. When enclosing in a container, we will need to give a Base name that will be used for all the created files. It is not required if we are not transferring files or enclosing them in a container.

A screenshot of a computer

Description automatically generated

In the Current command line tab, we can see the command line options being added or removed while configuring the UI. This Window will show more options in the command line as we add options. Please note that the destination path in your case will be different from the one shown in the image. Notice the --tflush flag here. It means that when this command line was created, the Flush checkbox was still checked.

A screenshot of a computer

Description automatically generated

By checking the Use Module Options checkbox, the right side of the KAPE Window will also be enabled.

A screenshot of a computer

Description automatically generated

When using both Target and Module Options, providing Module Source is not required. The selected Modules will use the Target destination as the source.

A screenshot of a login box

Description automatically generated

The rest of the options for Modules are similar to the ones for Targets, so we won't go into details for them.

Below you will see what the configuration looks like when we have KAPE all set up for collecting Targets and processing them using Modules.

A screenshot of a computer

Description automatically generated

Selected is the KapeTriage compound Target and !EZParser Compound Module. The command line below shows the CLI command that will be run. The Execute! button in the bottom right corner will execute the command. The Disable flush warnings checkbox underneath it will not warn us when we are using the Flush flags. When we press Execute! We will see a command line window open and show us the logs as KAPE performs its tasks. It will take a few minutes to execute since it will be collecting all the data and then running the module processes on it. Once it completes, it will show us the total execution time, and we can press any key to terminate the command window.

D:\Kape\kape.exe

KAPE version 1.1.0.1 Author: Eric Zimmerman (kape@kroll.com)

KAPE directory: D:\KAPE

Command line: --tsource C: --tdest C:\Users\Umair\Desktop\kape --target KapeTriage --mdest C:\Users\Umair\Desktop\4n6-2 --module !EZParser --gui

System info: Machine name: UMAIR-THINKBOOK, 64-bit: True, User: Umair OS: Windows10 (10.0.22000)

Using Target operations

Found 14 targets. Expanding targets to file list...

Target 'ApplicationEvents' with Id '2da16dbf-ea47-448e-a00f-fc442c3109ba' already processed. Skipping!

Target 'ApplicationEvents' with Id '2da16dbf-ea47-448e-a00f-fc442c3109ba' already processed. Skipping!

Target 'ApplicationEvents' with Id '2da16dbf-ea47-448e-a00f-fc442c3109ba' already processed. Skipping!

Target 'ApplicationEvents' with Id '2da16dbf-ea47-448e-a00f-fc442c3109ba' already processed. Skipping!

Target 'ApplicationEvents' with Id '2da16dbf-ea47-448e-a00f-fc442c3109ba' already processed. Skipping!

Found 3,059 files in 4.257 seconds. Beginning copy...

Deferring 'C:\Windows\System32\winevt\logs\Application.evtx' due to IOException...

Deferring 'C:\Windows\System32\winevt\Logs\Microsoft-Windows-Windows Defender%4Operational.evtx' due to IOException...

Deferring 'C:\Windows\System32\winevt\Logs\Microsoft-Windows-Windows Defender%4WHC.evtx' due to IOException...

Deferring 'C:\ProgramData\Microsoft\Windows Defender\Support\MPDetection-20220126-183133.log' due to IOException...

Deferring 'C:\ProgramData\Microsoft\Windows Defender\Support\MPDeviceControl-20211016-164735.log' due to IOException...

Deferring 'C:\ProgramData\Microsoft\Windows Defender\Support\MPLog-10172021-040927.log' due to IOException...

Deferring 'C:\ProgramData\Microsoft\Windows Defender\Support\MpWppTracing-20220210-070038-00000003-ffffffff.bin' due to IOException...

Deferring 'C:\Windows\System32\winevt\logs\HardwareEvents.evtx' due to IOException...

Deferring 'C:\Windows\System32\winevt\logs\IntelAudioServiceLog.evtx' due to IOException...

Deferring 'C:\Windows\System32\winevt\logs\Internet Explorer.evtx' due to IOException...

.

Executing remaining modules...

Running 'EvtxECmd\EvtxECmd.exe': -d C:\Users\Umair\Desktop\kape --csv C:\Users\Umair\Desktop\4n6-2\EventLogs

Running 'JLECmd.exe': -d C:\Users\Umair\Desktop\kape --csv C:\Users\Umair\Desktop\4n6-2\FileFolderAccess -q

Running 'LECmd.exe': -d C:\Users\Umair\Desktop\kape --csv C:\Users\Umair\Desktop\4n6-2\FileFolderAccess -q

Running 'PECmd.exe': -d C:\Users\Umair\Desktop\kape --csv C:\Users\Umair\Desktop\4n6-2\ProgramExecution -q

Running 'RBCmd.exe': -d C:\Users\Umair\Desktop\kape --csv C:\Users\Umair\Desktop\4n6-2\FileDeletion -q

Running 'RECmd\RECmd.exe': -d C:\Users\Umair\Desktop\kape --bn BatchExamples\Kroll\_Batch.reb --nl false --csv C:\Users\Umair\Desktop\4n6-2\Registry -q

Running 'SBECmd.exe': -d C:\Users\Umair\Desktop\kape --csv C:\Users\Umair\Desktop\4n6-2\FileFolderAccess -q

Running 'SQLECmd\SQLECmd.exe': -d C:\Users\Umair\Desktop\kape --csv C:\Users\Umair\Desktop\4n6-2\SQLDatabases

Running 'SrumECmd.exe': -d C:\Users\Umair\Desktop\kape -k --csv C:\Users\Umair\Desktop\4n6-2\SystemActivity

Running 'SumECmd.exe': -d C:\Users\Umair\Desktop\kape\Windows\System32\LogFiles\SUM --csv C:\Users\Umair\Desktop\4n6-2\SUMDatabase

Executed 18 processors in 192.2738 seconds

Total execution time: 258.1812 seconds

Press any key to exit

Notice that at the backend, KAPE is running the kape.exe in a command line. We can check out the files created by KAPE once it completes processing them. The below snapshot shows our Module destination. Notice how KAPE has processed the files according to different categories.

A screenshot of a computer

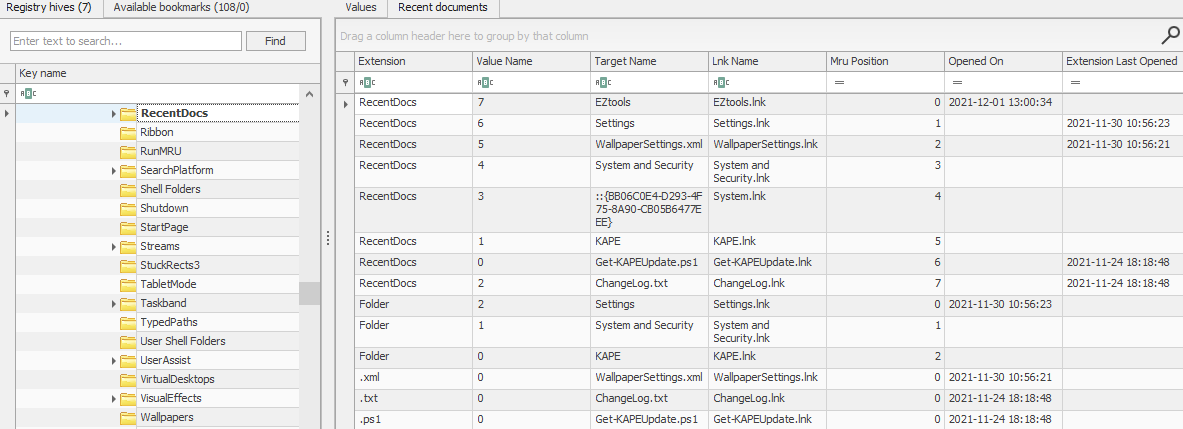
Description automatically generated

Let's collect triage data using the KAPETriage package, process it using !EZParser module,

**Recent Files:**

Windows maintains a list of recently opened files for each user. As we might have seen when using Windows Explorer, it shows us a list of recently used files. This information is stored in the NTUSER hive and can be found on the following location:

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs



Registry Explorer allows us to sort data contained in registry keys quickly. For example, the Recent documents tab arranges the Most Recently Used (MRU) file at the top of the list. Registry Explorer also arranges them so that the Most Recently Used (MRU) file is shown at the top of the list and the older ones later.

Another interesting piece of information in this registry key is that there are different keys with file extensions, such as .pdf, .jpg, .docx etc. These keys provide us with information about the last used files of a specific file extension. So if we are looking specifically for the last used PDF files, we can look at the following registry key:

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.pdf

Registry Explorer also lists the Last Opened time of the files. Answer Question # 1 by looking at the above screenshot.

**Office Recent Files:**

Similar to the Recent Docs maintained by Windows Explorer, Microsoft Office also maintains a list of recently opened documents. This list is also located in the NTUSER hive. It can be found in the following location:

NTUSER.DAT\Software\Microsoft\Office\VERSION

The version number for each Microsoft Office release is different. An example registry key will look like this:

NTUSER.DAT\Software\Microsoft\Office\15.0\Word

Here, the 15.0 refers to Office 2013. A list of different Office releases and their version numbers can be found on [this link](https://docs.microsoft.com/en-us/deployoffice/install-different-office-visio-and-project-versions-on-the-same-computer#office-releases-and-their-version-number).

Starting from Office 365, Microsoft now ties the location to the user's [live ID](https://www.microsoft.com/security/blog/2008/05/07/what-is-a-windows-live-id/). In such a scenario, the recent files can be found at the following location.

NTUSER.DAT\Software\Microsoft\Office\VERSION\UserMRU\LiveID\_####\FileMRU

In such a scenario, the recent files can be found at the following location. This location also saves the complete path of the most recently used files.

**ShellBags:**

When any user opens a folder, it opens in a specific layout. Users can change this layout according to their preferences. These layouts can be different for different folders. This information about the Windows *'shell'* is stored and can identify the Most Recently Used files and folders. Since this setting is different for each user, it is located in the user hives. We can find this information on the following locations:

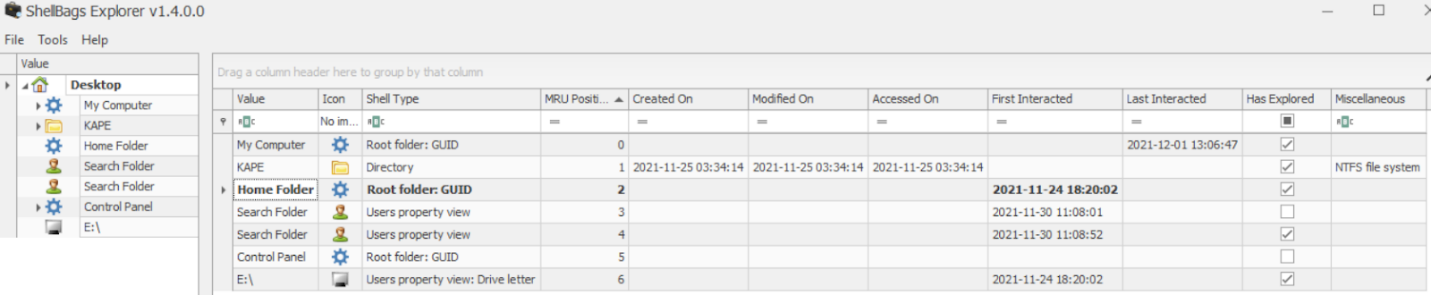
USRCLASS.DAT\Local Settings\Software\Microsoft\Windows\Shell\Bags

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

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

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

Registry Explorer doesn't give us much information about ShellBags. However, another tool from Eric Zimmerman's tools called the ShellBag Explorer shows us the information in an easy-to-use format. We just have to point to the hive file we have extracted, and it parses the data and shows us the results. An example is shown below. Take a look and answer Question # 2.



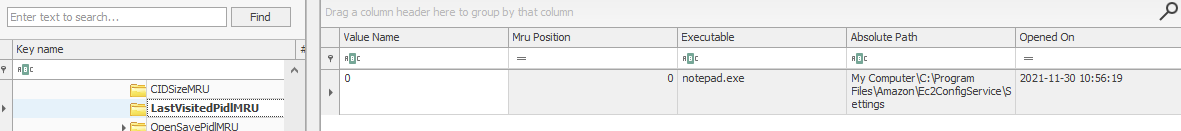
**Open/Save and LastVisited Dialog MRUs:**

When we open or save a file, a dialog box appears asking us where to save or open that file from. It might be noticed that once we open/save a file at a specific location, Windows remembers that location. This implies that we can find out recently used files if we get our hands on this information. We can do so by examining the following registry keys

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePIDlMRU

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPidlMRU

This is how Registry Explorer shows this registry key. Take a look to answer Question # 3 and 4.



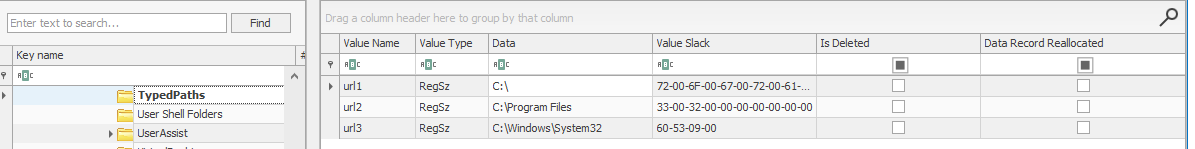
**Windows Explorer Address/Search Bars:**

Another way to identify a user's recent activity is by looking at the paths typed in the Windows Explorer address bar or searches performed using the following registry keys, respectively.

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\WordWheelQuery

Here is how the TypedPaths key looks like in Registry Explorer:



**UserAssist**:

Windows keeps track of applications launched by the user using Windows Explorer for statistical purposes in the User Assist registry keys. These keys contain information about the programs launched, the time of their launch, and the number of times they were executed. However, programs that were run using the command line can't be found in the User Assist keys. The User Assist key is present in the NTUSER hive, mapped to each user's GUID. We can find it at the following location:

NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count

Take a look at the below screenshot from Registry Explorer and answer Question #1.

A screenshot of a computer

Description automatically generated

**ShimCache:**

ShimCache is a mechanism used to keep track of application compatibility with the OS and tracks all applications launched on the machine. Its main purpose in Windows is to ensure backward compatibility of applications. It is also called Application Compatibility Cache (AppCompatCache). It is located in the following location in the SYSTEM hive:

SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache

ShimCache stores file name, file size, and last modified time of the executables.

Our goto tool, the Registry Explorer, doesn't parse ShimCache data in a human-readable format, so we go to another tool called AppCompatCache Parser, also a part of Eric Zimmerman's tools. It takes the SYSTEM hive as input, parses the data, and outputs a CSV file that looks like this:

A screenshot of a computer

Description automatically generated

We can use the following command to run the AppCompatCache Parser Utility:

AppCompatCacheParser.exe --csv <path to save output> -f <path to SYSTEM hive for data parsing> -c <control set to parse>

The output can be viewed using EZviewer, another one of Eric Zimmerman's tools.

**AmCache:**

The AmCache hive is an artifact related to ShimCache. This performs a similar function to ShimCache, and stores additional data related to program executions. This data includes execution path, installation, execution and deletion times, and SHA1 hashes of the executed programs. This hive is located in the file system at:

C:\Windows\appcompat\Programs\Amcache.hve

Information about the last executed programs can be found at the following location in the hive:

Amcache.hve\Root\File\{Volume GUID}\

This is how Registry Explorer parses the AmCache hive:

A blurry image of a computer screen

Description automatically generated

**BAM/DAM:**

Background Activity Monitor or BAM keeps a tab on the activity of background applications. Similar Desktop Activity Moderator or DAM is a part of Microsoft Windows that optimizes the power consumption of the device. Both of these are a part of the Modern Standby system in Microsoft Windows.

In the Windows registry, the following locations contain information related to BAM and DAM. This location contains information about last run programs, their full paths, and last execution time.

SYSTEM\CurrentControlSet\Services\bam\UserSettings\{SID}

SYSTEM\CurrentControlSet\Services\dam\UserSettings\{SID}

Below you can see how Registry Explorer parses data from BAM:

A screenshot of a computer

Description automatically generated

When performing forensics on a machine, often the need arises to identify if any USB or removable drives were attached to the machine. If so, any information related to those devices is important for a forensic investigator. In this task, we will go through the different ways to find information on connected devices and the drives on a system using the registry.

**Device identification:**

The following locations keep track of USB keys plugged into a system. These locations store the vendor id, product id, and version of the USB device plugged in and can be used to identify unique devices. These locations also store the time the devices were plugged into the system.

SYSTEM\CurrentControlSet\Enum\USBSTOR

SYSTEM\CurrentControlSet\Enum\USB

Registry Explorer shows this information in a nice and easy-to-understand way. Take a look at this and answer Questions # 1 and 2.

A screenshot of a computer

Description automatically generated

**First/Last Times:**

Similarly, the following registry key tracks the first time the device was connected, the last time it was connected and the last time the device was removed from the system.

SYSTEM\CurrentControlSet\Enum\USBSTOR\Ven\_Prod\_Version\USBSerial#\Properties\{83da6326-97a6-4088-9453-a19231573b29}\####

In this key, the #### sign can be replaced by the following digits to get the required information:

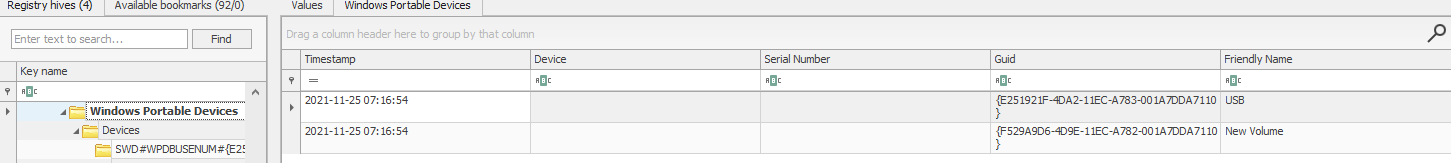
|  |  |
| --- | --- |
| **Value** | **Information** |
| 0064 | First Connection time |
| 0066 | Last Connection time |
| 0067 | Last removal time |

Although we can check this value manually, as we have seen above, Registry Explorer already parses this data and shows us if we select the USBSTOR key.

**USB device Volume Name:**

The device name of the connected drive can be found at the following location:

SOFTWARE\Microsoft\Windows Portable Devices\Devices

We can compare the GUID we see here in this registry key and compare it with the Disk ID we see on keys mentioned in device identification to correlate the names with unique devices. Take a look at these two screenshots and answer Question # 3.

Combining all of this information, we can create a fair picture of any USB devices that were connected to the machine we're investigating.

**Windows Prefetch files**

When a program is run in Windows, it stores its information for future use. This stored information is used to load the program quickly in case of frequent use. This information is stored in prefetch files which are located in the C:\Windows\Prefetch directory.

Prefetch files have an extension of .pf. Prefetch files contain the last run times of the application, the number of times the application was run, and any files and device handles used by the file. Thus it forms an excellent source of information about the last executed programs and files.

We can use Prefetch Parser (PECmd.exe) from Eric Zimmerman's tools for parsing Prefetch files and extracting data. When we run PECmd.exe in an elevated command prompt, we get this output:

Administrator: Command Prompt

user@machine$ PECmd.exe

PECmd version 1.4.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)

https://github.com/EricZimmerman/PECmd

d Directory to recursively process. Either this or -f is required

f File to process. Either this or -d is required

k Comma separated list of keywords to highlight in output. By default, 'temp' and 'tmp' are highlighted. Any additional keywords will be added to these.

o When specified, save prefetch file bytes to the given path. Useful to look at decompressed Win10 files

q Do not dump full details about each file processed. Speeds up processing when using --json or --csv. Default is FALSE

json Directory to save json representation to.

jsonf File name to save JSON formatted results to. When present, overrides default name

csv Directory to save CSV results to. Be sure to include the full path in double quotes

csvf File name to save CSV formatted results to. When present, overrides default name

html Directory to save xhtml formatted results to. Be sure to include the full path in double quotes

dt The custom date/time format to use when displaying timestamps. See https://goo.gl/CNVq0k for options. Default is: yyyy-MM-dd HH:mm:ss

mp When true, display higher precision for timestamps. Default is FALSE

vss Process all Volume Shadow Copies that exist on drive specified by -f or -d . Default is FALSE

dedupe Deduplicate -f or -d & VSCs based on SHA-1. First file found wins. Default is TRUE

debug Show debug information during processing

trace Show trace information during processing

Examples: PECmd.exe -f "C:\Temp\CALC.EXE-3FBEF7FD.pf"

PECmd.exe -f "C:\Temp\CALC.EXE-3FBEF7FD.pf" --json "D:\jsonOutput" --jsonpretty

PECmd.exe -d "C:\Temp" -k "system32, fonts"

PECmd.exe -d "C:\Temp" --csv "c:\temp" --csvf foo.csv --json c:\temp\json

PECmd.exe -d "C:\Windows\Prefetch"

Short options (single letter) are prefixed with a single dash. Long commands are prefixed with two dashes

Either -f or -d is required. Exiting

To run Prefetch Parser on a file and save the results in a CSV, we can use the following command:

PECmd.exe -f <path-to-Prefetch-files> --csv <path-to-save-csv>

Similarly, for parsing a whole directory, we can use the following command:

PECmd.exe -d <path-to-Prefetch-directory> --csv <path-to-save-csv>

We can use this information to answer the questions at the end.

**Windows 10 Timeline**

Windows 10 stores recently used applications and files in an SQLite database called the Windows 10 Timeline. This data can be a source of information about the last executed programs. It contains the application that was executed and the focus time of the application. The Windows 10 timeline can be found at the following location:

C:\Users\<username>\AppData\Local\ConnectedDevicesPlatform\{randomfolder}\ActivitiesCache.db

We can use Eric Zimmerman's WxTCmd.exe for parsing Windows 10 Timeline. We get the following options when we run it:

Administrator: Command Prompt

user@machine$ WxTCmd.exe

WxTCmd version 0.6.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)

https://github.com/EricZimmerman/WxTCmd

f File to process. Required

csv Directory to save CSV formatted results to. Be sure to include the full path in double quotes

dt The custom date/time format to use when displaying timestamps. See https://goo.gl/CNVq0k for options. Default is: yyyy-MM-dd HH:mm:ss

Examples: WxTCmd.exe -f "C:\Users\eric\AppData\Local\ConnectedDevicesPlatform\L.eric\ActivitiesCache.db" --csv c:\temp

Database files are typically found at 'C:\Users\\AppData\Local\ConnectedDevicesPlatform\L.\ActivitiesCache.db'

Short options (single letter) are prefixed with a single dash. Long commands are prefixed with two dashes

-f is required. Exiting

We can use the following command to run WxTCmd:

WxTCmd.exe -f <path-to-timeline-file> --csv <path-to-save-csv>

**Windows Jump Lists**

Windows introduced jump lists to help users go directly to their recently used files from the taskbar. We can view jumplists by right-clicking an application's icon in the taskbar, and it will show us the recently opened files in that application. This data is stored in the following directory:

C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations

Jumplists include information about the applications executed, first time of execution, and last time of execution of the application against an AppID.

We can use Eric Zimmerman's JLECmd.exe to parse Jump Lists. We get the following options when we run it:

Administrator: Command Prompt

user@machine$ JLECmd.exe

JLECmd version 1.4.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)

https://github.com/EricZimmerman/JLECmd

d Directory to recursively process. Either this or -f is required

f File to process. Either this or -d is required

q Only show the filename being processed vs all output. Useful to speed up exporting to json and/or csv. Default is FALSE

all Process all files in directory vs. only files matching \*.automaticDestinations-ms or \*.customDestinations-ms. Default is FALSE

csv Directory to save CSV formatted results to. Be sure to include the full path in double quotes

csvf File name to save CSV formatted results to. When present, overrides default name

html Directory to save xhtml formatted results to. Be sure to include the full path in double quotes

json Directory to save json representation to. Use --pretty for a more human readable layout

pretty When exporting to json, use a more human readable layout. Default is FALSE

ld Include more information about lnk files. Default is FALSE

fd Include full information about lnk files (Alternatively, dump lnk files using --dumpTo and process with LECmd). Default is FALSE

appIds Path to file containing AppIDs and descriptions (appid|description format). New appIds are added to the built-in list, existing appIds will have their descriptions updated

dumpTo Directory to save exported lnk files

withDir When true, show contents of Directory not accounted for in DestList entries

Debug Debug mode

dt The custom date/time format to use when displaying timestamps. See https://goo.gl/CNVq0k for options. Default is: yyyy-MM-dd HH:mm:ss

mp Display higher precision for timestamps. Default is FALSE

Examples: JLECmd.exe -f "C:\Temp\f01b4d95cf55d32a.customDestinations-ms" --mp

JLECmd.exe -f "C:\Temp\f01b4d95cf55d32a.automaticDestinations-ms" --json "D:\jsonOutput" --jsonpretty

JLECmd.exe -d "C:\CustomDestinations" --csv "c:\temp" --html "c:\temp" -q

JLECmd.exe -d "C:\Users\e\AppData\Roaming\Microsoft\Windows\Recent" --dt "ddd yyyy MM dd HH:mm:ss.fff"

Short options (single letter) are prefixed with a single dash. Long commands are prefixed with two dashes

Either -f or -d is required. Exiting

We can use the following command to parse Jumplists using JLECmd.exe:

JLECmd.exe -f <path-to-Jumplist-file> --csv <path-to-save-csv>

In the folder named triage, present on the Desktop of the attached machine, we have extracted the Windows directory of a system we want to investigate. It retains the directory structure of the original Windows directory, that is, C:\Windows directory from the system is mapped on to C:\users\thm-4n6\Desktop\triage\C\Windows. Now let's use the information we have learned to perform analysis on the data saved in the folder named triage on the Desktop in the attached VM and answer the following questions.

If you are having trouble viewing the CSV file, you can use EZviewer from the EZtools folder.

**Shortcut Files**

Windows creates a shortcut file for each file opened either locally or remotely. The shortcut files contain information about the first and last opened times of the file and the path of the opened file, along with some other data. Shortcut files can be found in the following locations:

C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\

C:\Users\<username>\AppData\Roaming\Microsoft\Office\Recent\

We can use Eric Zimmerman's LECmd.exe (Lnk Explorer) to parse Shortcut files. When we run the LECmd.exe, we see the following options:

Administrator: Command Prompt

user@machine$ LECmd.exe

LECmd version 1.4.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)

https://github.com/EricZimmerman/LECmd

d Directory to recursively process. Either this or -f is required

f File to process. Either this or -d is required

q Only show the filename being processed vs all output. Useful to speed up exporting to json and/or csv. Default is FALSE

r Only process lnk files pointing to removable drives. Default is FALSE

all Process all files in directory vs. only files matching \*.lnk. Default is FALSE

csv Directory to save CSV formatted results to. Be sure to include the full path in double quotes

csvf File name to save CSV formatted results to. When present, overrides default name

xml Directory to save XML formatted results to. Be sure to include the full path in double quotes

html Directory to save xhtml formatted results to. Be sure to include the full path in double quotes

json Directory to save json representation to. Use --pretty for a more human readable layout

pretty When exporting to json, use a more human readable layout. Default is FALSE

nid Suppress Target ID list details from being displayed. Default is FALSE

neb Suppress Extra blocks information from being displayed. Default is FALSE

dt The custom date/time format to use when displaying time stamps. See https://goo.gl/CNVq0k for options. Default is: yyyy-MM-dd HH:mm:ss

mp Display higher precision for time stamps. Default is FALSE

Examples: LECmd.exe -f "C:\Temp\foobar.lnk"

LECmd.exe -f "C:\Temp\somelink.lnk" --json "D:\jsonOutput" --jsonpretty

LECmd.exe -d "C:\Temp" --csv "c:\temp" --html c:\temp --xml c:\temp\xml -q

LECmd.exe -f "C:\Temp\some other link.lnk" --nid --neb

LECmd.exe -d "C:\Temp" --all

Short options (single letter) are prefixed with a single dash. Long commands are prefixed with two dashes

Either -f or -d is required. Exiting

We can use the following command to parse shortcut files using LECmd.exe:

LECmd.exe -f <path-to-shortcut-files> --csv <path-to-save-csv>

The creation date of the shortcut file points to the date/time when the file was first opened. The date/time of modification of the shortcut file points to the last time the file was accessed.

**IE/Edge history**

An interesting thing about the IE/Edge browsing history is that it includes files opened in the system as well, whether those files were opened using the browser or not. Hence, a valuable source of information on opened files in a system is the IE/Edge history. We can access the history in the following location:

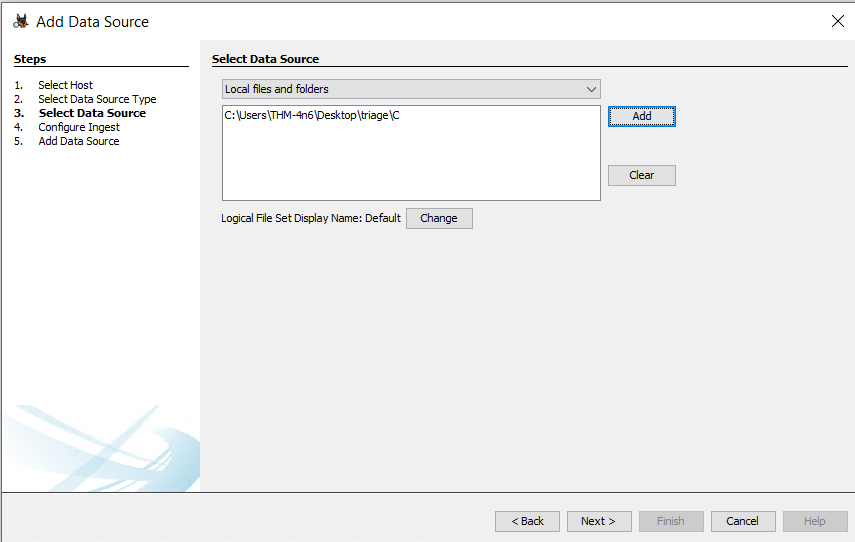
C:\Users\<username>\AppData\Local\Microsoft\Windows\WebCache\WebCacheV\*.dat

The files/folders accessed appear with a file:///\* prefix in the IE/Edge history. Though several tools can be used to analyze Web cache data, you can use Autopsy to do so in the attached VM. For doing that, select Logical Files as a data source.

A screenshot of a computer

Description automatically generated

It will then ask you to select the path from which you want files to be analyzed. You can provide the path to the triage folder.



In the Window where Autopsy asks about ingest modules to process data, check the box in front of 'Recent Activity' and uncheck everything else.

A screenshot of a computer

Description automatically generated

You will be able to view local files accessed in the Web history option in the left panel.

A screenshot of a computer

Description automatically generated

This is what it will look like in the right panel.

A screenshot of a computer

Description automatically generated

As shown above, the 'Data Artifacts' tab displays information about the file accessed.

**Jump Lists**

As we already learned in the last task, Jump Lists create a list of the last opened files. This information can be used to identify both the last executed programs and the last opened files in a system. Remembering from the last task, Jump Lists are present at the following location:

C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations

Setupapi dev logs for USB devices

When any new device is attached to a system, information related to the setup of that device is stored in the setupapi.dev.log. This log is present at the following location:

C:\Windows\inf\setupapi.dev.log

This log contains the device serial number and the first/last times when the device was connected.

A screenshot of a computer screen

Description automatically generated

Here is what it looks like when opened in Notepad.exe. Notice the first line where we can see the device ID and Serial Number.

## Shortcut files

As we learned in the previous task, shortcut files are created automatically by Windows for files opened locally or remotely. These shortcut files can sometimes provide us with information about connected USB devices. It can provide us with information about the volume name, type, and serial number. Recalling from the previous task, this information can be found at:

C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\

C:\Users\<username>\AppData\Roaming\Microsoft\Office\Recent\