MEMORY ANALYSIS FOR FUN AND PROFIT

HADESS

WWW.HADESS.IO

Deep Dive Into Memory For Fun & Profit

Delve into the thrilling realm of memory analysis where each byte tells a story. This comprehensive guide explores the fascinating techniques of memory forensics, a field where the ephemeral traces of digital interactions are captured and decoded. From unearthing incognito browsing sessions to dissecting malware footprints, discover how experts turn volatile memory into a treasure trove of information. Whether for cybersecurity, legal investigations, or academic research, this article unveils the power and potential of memory analysis, turning transient data into valuable insights.

ID	Attacks	Mem Address/Keys	Commands	Example
1	Process Injection	Process memory space	`volatility -f memdump.img --profile=Win10x64 pslist`	Malware injects code into a legitimate process; detected via Volatility pslist.
2	Rootkit Detection	Kernel memory	`volatility -f memdump.img --profile=Win10x64 malfind`	Hidden rootkit in kernel memory; revealed by malfind plugin.
3	Credential Dumping	LSASS process memory	`volatility -f memdump.img --profile=Win10x64 ldrmodules`	Extracting passwords from LSASS; detected via ldrmodules plugin.
4	Network Spoofing	Network stack memory	`volatility -f memdump.img --profile=Win10x64 netscan`	Spoofed network connections; netscan shows irregular connections.
5	Keylogger Detection	Keyboard buffer	Custom scripts/commands based on anomaly detection	Keylogging activity; detected by analyzing unusual keyboard interrupts.
6	Fileless Malware	System RAM	`volatility -f memdump.img --profile=Win10x64 malfind`	Malware executing directly in memory; malfind shows suspicious memory segments.
7	Code Injection in Browsers	Browser process memory	`volatility -f memdump.img --profile=Win10x64 --plugin=chromehistory`	Injected script in browser process; detected via browser history plugin.
8	Stealthy Malware via PEB	Process Environment Block	`volatility -f memdump.img --profile=Win10x64 --plugin=check_peb_spoof`	Malware altering PEB; detected with check_peb_spoof plugin.
9	Suspicious Parent-Child Process Relationship	Process metadata	`volatility -f memdump.img --profile=Win10x64 --plugin=check_parent_spoof`	Unusual parent process; detected by check_parent_spoof plugin.
10	Unauthorized Remote Access	Remote desktop process memory	`volatility -f memdump.img --profile=Win10x64 --plugin=anydesk`	Unauthorized AnyDesk access; detected via AnyDesk plugin.

Clipboard is Everything

The copy-paste functionality in Windows is a fundamental feature used for transferring data, especially text and strings, between applications or within an application. It utilizes the Windows Clipboard, a special location in memory where data is temporarily stored during the copy-paste

Clipboard in Windows

Basic Operation

- **Copy:** When text is copied, it is stored in the Clipboard.
- **Paste:** Pasting retrieves the data from the Clipboard and inserts it into the target location.

Data Formats

- The Clipboard can store data in multiple formats, such as plain text, rich text, HTML, and others.

Tools for Clipboard Management

1. **Clipboard Viewer**: A utility to view the contents of the Clipboard.
2. **Clip.exe**: A command-line tool introduced in later versions of Windows for redirecting output to the Clipboard.
3. **PowerShell Cmdlets**: For advanced Clipboard operations, like accessing history.

Accessing Clipboard History

- Windows 10 (Version 1809 and later) introduced a Clipboard history feature, allowing users to access a history of items that were copied.

Enabling Clipboard History

- Use `Win + V` to open Clipboard history.
- Enable it in system settings: `Settings` > `System` > `Clipboard` > `Clipboard History`.

Accessing History via Code

- Use Windows APIs or PowerShell scripts to programmatically access Clipboard history.

Commands and Code Snippets

For Basic Clipboard Operations

- **Copy**: `Ctrl + C` or `Right-click > Copy`
- **Paste**: `Ctrl + V` or `Right-click > Paste`

PowerShell Command to Access Clipboard

`Get-Clipboard # Retrieves the current Clipboard content. Set-Clipboard # Sets or clears the Clipboard content.`

Using Clip.exe in Command Prompt

`dir | clip # Copies the output of 'dir' command to Clipboard.`

Clipboard in Different Windows Versions

Windows XP

- Basic Clipboard functionality.
- No built-in history or advanced management tools.

Windows 7

- Introduced Clip.exe for command-line operations.

Windows 10 (Build 1809 and later)

- Clipboard history feature added.
- Enhanced Clipboard settings in system settings.

Windows 11

- Continued improvements in Clipboard history and synchronization features.

Windows 10 (Build 1809 and later)

Clipboard History Storage

- Clipboard data is temporarily stored in memory.
- History is stored in `%LOCALAPPDATA%\Microsoft\Windows\Clipboard`.

Clipboard History Duration

- Items remain in the Clipboard history until the system is restarted or the history is manually cleared.

Advanced Tools

- `Get-Clipboard` and `Set-Clipboard` in PowerShell for programmatic access.
- Third-party tools like Ditto for extended clipboard functionalities.

Accessing Clipboard History

`Get-Clipboard -Format Text -TextFormatType UnicodeText # Fetches clipboard content in a specific format.`

Windows 11

- Similar to Windows 10 with additional synchronization features via cloud.

macOS Clipboard Management

Clipboard Storage

- Data copied to the clipboard is stored in memory.

Clipboard History Duration

- macOS does not natively support clipboard history; items are replaced upon new copy action.

Advanced Tools

- Third-party applications like Paste or CopyClip for clipboard history management.

Accessing Clipboard Data

- Use AppleScript or Automator for advanced clipboard operations.

```
set the clipboard to "Your text"  # Sets the clipboard content.
get the clipboard  # Retrieves the current clipboard content.
```

Linux Clipboard Management

X Window System (X11)

Clipboard Storage

- X11 has multiple clipboards: PRIMARY, SECONDARY, and CLIPBOARD.
- Data is stored in memory.

Clipboard History Duration

- Typically, clipboard managers like `xclip` or `Klipper` are used to maintain history.

Advanced Clipboard Management

`xclip -selection clipboard -o # Outputs the content of the CLIPBOARD. echo "Text" | xclip -selection clipboard # Copies text to the CLIPBOARD.`

Wayland

- Tools like `wl-clipboard` are used for clipboard operations.

`wl-copy < "Text file.txt" # Copies file content to clipboard. wl-paste > "Output file.txt" # Pastes clipboard content to a file.`

Clipboard History Location and Duration

- **Windows**: `%LOCALAPPDATA%\Microsoft\Windows\Clipboard`, until system restart.
- **macOS**: No native history; third-party apps determine storage.
- **Linux (X11)**: Managed by clipboard managers; varies by tool.

Insights into Browser Incognito Mode and Memory Forensics

This guide delves into the advanced aspects of browser incognito modes across various browsers and operating systems, focusing on memory forensics and the extraction of browsing data. We cover the specifics for Chrome, Firefox, and Edge on Windows, macOS, and Linux, detailing the memory locations, forensic tools, and techniques for data retrieval.

Windows

Google Chrome Incognito

- **Memory Location**: Volatile; data resides in RAM during active session.
- **Forensic Tools & Commands**:
  - **WinHex** or **Magnet RAM Capture** for RAM imaging.
  - **Volatility** framework for analysis: `volatility -f [MEMORY DUMP] --profile=[PROFILE] chrome`.

Mozilla Firefox Incognito

- **Memory Location**: Similar to Chrome, data held in RAM.
- **Forensic Approach**:
  - Capture RAM using tools like **FTK Imager**.
  - Analyze with **Volatility**: `volatility -f [MEMORY DUMP] --profile=[PROFILE] firefox`.

Microsoft Edge Incognito

- **Memory Location**: Stored in RAM, similar to Chrome.
- **Forensic Methodology**:
  - Use **Belkasoft RAM Capturer**.
  - Analyze with **Volatility** or **Belkasoft Evidence Center**.

Fetching Incognito Data

- **Command**: `.\winpmem_mini_x64.exe -o memorydump.raw`

macOS

All Browsers (Chrome, Firefox, Edge)

- **Memory Location**: Data in RAM, purged after session ends.
- **Forensic Tools**:
  - **MacQuisition** for live data acquisition.
  - **BlackLight** for analysis.

Fetching Incognito Data

- **Command**: `sudo /usr/bin/hdiutil create -srcdevice /dev/diskX memdump.dmg`

Linux

All Browsers (Chrome, Firefox, Edge)

- **Memory Location**: Resides in system RAM.
- **Forensic Tools**:
  - **LiME** (Linux Memory Extractor) for RAM capture.
  - Analyze with tools like **Volatility**.

- **Command**: `sudo lime-forensics-dkms /proc/lime_mem.mem "path=tcp format=lime timeout=0"`

uninstallinfo.py

Scenario

You are investigating a corporate environment where a suspected data leak occurred through unauthorized software. Your task is to identify if any suspicious software was installed and then removed from a company computer.

Using `uninstallinfo.py`

- **Prerequisite**: You have a memory dump (`comp_memory.dmp`) from the suspect's Windows machine.
- **Volatility Profile**: Identified as `Win2012R2x64` (example profile).

Commands and Analysis

1. **Load the Plugin**:

   ```
   volatility --plugins=[path_to_plugin_folder] -f comp_memory.dmp --profile=Win2012R2x64 uninstallinfo
   ```
2. **Output**: The plugin scans the memory for traces of uninstalled programs, specifically looking at Windows Registry keys related to installed applications.
3. **Sample Output**:

   ```
   Program Name: SuspiciousVPNClient Uninstall Key: HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall\SuspiciousVPN Install Date: 2022-05-15 Uninstall Date: 2022-05-20
   ```
4. **Interpretation**: This output indicates that `SuspiciousVPNClient` was installed and then uninstalled within a short timeframe, aligning with the suspected data leak period.

2. firefoxhistory.py

Scenario

In a legal investigation, you need to determine if a suspect accessed specific websites that could be related to the case, even if they used Firefox's private browsing mode.

Using `firefoxhistory.py`

- **Prerequisite**: Memory dump (`user_memory.dmp`) from the suspect's Linux-based computer.
- **Volatility Profile**: Identified as `LinuxUbuntu20x64` (example profile).

Commands and Analysis

1. **Load the Plugin**:

   ```
   volatility --plugins=[path_to_plugin_folder] -f user_memory.dmp --profile=LinuxUbuntu20x64 firefoxhistory
   ```
2. **Output**: The plugin searches for instances of Firefox processes and extracts browsing history, even from private sessions.
3. **Sample Output**:

   ```
   URL: http://example-sensitive-website.com Title: Sensitive Content Last Visit Time: 2022-07-05 10:23:45 Visit Count: 3
   ```
4. **Interpretation**: This data suggests the suspect visited a sensitive website multiple times, which could be crucial for the investigation.

3. chromehistory.py

Purpose: Extracts browsing history from Google Chrome.

Scenario: Investigating a corporate espionage case where the suspect might have used Chrome to access sensitive information.

Usage:

- Load the plugin with: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] chromehistory`
- The plugin scans memory addresses where Chrome process artifacts might reside,

4. malfinddeep.py

Purpose: Identifies potentially malicious code segments hidden in the process memory space.

Scenario: Analyzing a compromised system to identify hidden or injected malware.

Usage:

- Run: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] malfinddeep`
- It investigates memory segments for anomalies, providing memory addresses of suspicious code.

5. openssh_sessionkeys

Purpose: Recovers OpenSSH session keys from memory.

Scenario: Retrieving SSH session keys in an incident response to a data breach.

Usage:

- Execute: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] openssh_sessionkeys`
- The plugin searches for specific memory patterns associated with SSH keys.

6. prefetch

Purpose: Analyzes Windows Prefetch files from memory, which can provide information on executed programs.

Scenario: Determining program execution history on a suspect's Windows machine.

Usage:

- Command: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] prefetch`
- Extracts Prefetch file details like application name and last run time.

7. passwordmanagers

Purpose: Extracts information from memory related to password manager applications.

Scenario: Investigating a data leak potentially caused by compromised password manager data.

Usage:

- Use: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] passwordmanagers`
- Searches for memory artifacts related to popular password managers.

8. richheader

Purpose: Analyzes the Rich Header in executables loaded in memory, useful for identifying the compiler and linker versions.

Scenario: Assessing if a piece of malware was compiled using a specific toolset.

Usage:

- Run: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] richheader`
- Extracts Rich Header data from executable files in memory.

9. zoneid

Purpose: Extracts the Zone.Identifier stream from memory, which can indicate the origin of downloaded files.

Scenario: Tracing the source of downloaded malware or suspicious files on a system.

Usage:

- Command: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] zoneid`
- Identifies the security zone from which a file was downloaded.

Purpose: Searches for API function calls in memory, useful for analyzing malware behavior.

Scenario: Investigating the behavior of a suspected malware by looking at its API calls.

Usage:

- Execute: `volatility -f [MEMORY_DUMP] --profile=[PROFILE] --plugins=[PLUGIN_PATH] apisearch`
- Finds memory addresses where specific API calls are made.

11. evtx.py (EVTX Log Extraction)

- **Functionality:** This plugin is designed to extract Windows Event Log (EVTX) records from memory dumps. It scans the memory dump for EVTX chunks and extracts records from these chunks.
- **Methodology:**
  - **Memory Mapping:** It uses `mmap` for memory mapping the dump file for efficient access.
  - **Chunk Identification:** Finds EVTX chunks within the mapped memory.
  - **Record Extraction:** Extracts individual records from these chunks and decodes them into XML format.
  - **Data Parsing:** Parses the XML to extract meaningful data like event IDs, channel, process IDs, etc.
- **Example Use Case:** This can be used in forensic investigations to retrieve event logs that provide insights into system activities, errors, and security-related events that occurred prior to the memory dump.

12. CryptoScan

- **Functionality:** CryptoScan scans memory for cryptocurrency transactions, particularly focusing on Bitcoin (BTC), Ethereum (ETH), and Ripple (XRP).
- **Methodology:**
  - **Process Analysis:** Targets specific process IDs (PIDs) for analysis.
  - **Transaction Signature Scanning:** Looks for patterns or signatures in memory that are indicative of cryptocurrency transactions.
- **Example Use Case:** In cases of financial fraud investigations or tracing illegal transactions, this plugin can be used to find evidence of cryptocurrency usage on a system.

13. check_spoof

- **Functionality:** `Check_peb_spoof` and `Check_parent_spoof` are designed to detect process spoofing by comparing PEB (Process Environment Block) data or analyzing parent-child process relationships.
- **Methodology:**
  - **PEB Analysis (check_peb_spoof):** Compares the process name in the PEB against the process name in the EPROCESS structure to detect mismatches.
  - **Parent-Child Relationship Analysis (check_parent_spoof):** Identifies discrepancies in parent-child process relationships that could indicate spoofing.
- **Example Use Case:** These plugins are crucial in identifying malware or processes that are attempting to disguise themselves as legitimate system processes.

14. anydesk.py (AnyDesk Artifact Analysis)

- **Functionality:** Parses artifacts related to the AnyDesk remote desktop application.
- **Methodology:**
  - **File Scanning:** Identifies and parses AnyDesk trace files (`ad.trace` , `ad_svc.trace`) in the memory dump.
  - **Data Extraction:** Extracts and decodes log entries from these files to provide insights into AnyDesk usage.
- **Example Use Case:** Useful in investigations involving unauthorized remote access where AnyDesk might have been used.

15. keepass.py (KeePass Password Recovery)

- **Functionality:** Attempts to recover potential password matches from the KeePass password manager.
- **Methodology:**
  - **Process Targeting:** Focuses on the KeePass process, identified by PID.
  - **Memory Analysis:** Scans the process memory for patterns that could represent

    - **Pattern Matching:** Utilizes a pattern matching approach to construct potential passwords from memory data.
- **Example Use Case:** In forensic scenarios where access to password-protected information is necessary, this plugin can help in recovering passwords from KeePass.

References

- https://github.com/volatilityfoundation/community3
- https://github.com/ZarKyo/awesome-volatility
- https://github.com/f-block/volatility-plugins