Memory Forensics

Lakshit Verma

April 10, 2024

Contents

1 Introduction

Memory Forensics is a type of digital forensics involving the collection and analysis of a system's memory, typically stored in `.mem` or `.img` files. We start by introducting the program we will use for Memory Forensics — **Volatility**, of which I specifically used **Volatility 3**.

The first command to run on a memory dump is `imageinfo`. We run this as `vol -f example.mem imageinfo`. This command scans the dump to find its memory profile, a.k.a the operating system and version it dumped. In the video, the examplee given is `Win10x64_14393`.

Now that we have our profile name, we can begin examining the processes that were running during the time the memory was being analysed. We can do this with three commands, `pslist`, `psscan` and `pstree`.

- `pslist`: It simply shows us all the processes the system finds to be running. Windows uses a doubly-linked-list data structure to store this, with one process linking to the subsequent and previous process, and so on.
- `psscan`: This performs a more in-depth search of it, searching for process blocks in all of memory, not trusting the linked list. This is because many malicious programs hide themselves from the former command by unlinking themselves from this linked list, and can only be found through this command.
- `pstree`: It shows all processes in a hierarchy of sorts, with parent and child processes clearly delineated.

A common way malware can be detected through memory forensics is through `svchost.exe`, a process running multiple instances on Windows machines. It should **always** have a parent process called `services.exe`, and anything otherwise is a large indicator of it being malware.

To dump any application running, we do `vol -f example.mem --profile=Windows_14393 procdump -p XXXX`, where `XXXX` is the process id (Pid) of the application we want to find.

Other commands include `memdump`, which dumps the raw memory data for the selected Pid and `modscan`, showing any drivers that were hidden or unloaded. For network analysis, we have `netscan`, which provides an extensive overview of all connections incoming or outgoing in the system.

2 Windows Example: Stuxnet

In this, we take an example of a memory dump of a Windows system infected with the infamous **Stuxnet**, and analyse it to find the malicious program.

Running `procdump` on the vmem file and glancing at `svchost.exe`, we find it to be harmless at face value. The first sign of suspicion comes from the fact that there are multiple instances of the program `lsass.exe`, the program handling all authentication processes on the system. Any more than one instance of this is grounds for a malware infection. One has a PPid (Parent Process ID) of 624, while two have it as 668. We then use the command `malfind` on the processes in question. We find the process with Pid 868 to have the permission `PAGE_EXECUTE_REWRITE`, which gives executable permissions. However, there is no file for it to execute, which is highly unusual behaviour.

We then use the command `hollowfind`, which is used to search for processes that have been "hollowed", meaning they have been duplicated, suspended, had malicious code injected into it, and then resumed. This gives us similar results to when we used `malfind`. Now, we extract both these suspicious programs (wth Pids of 868 and 1928), extracting the malicious executables. Taking their **sha256** hashes and running them through VirusTotal, we find these programs to indeed be **Stuxnet**.

3 Redline

Redline is a GUI based Memory Analysis tool. It has three types of memory collectors, **Standard**, **Comprehensive**, and **IOC Search**.

The Comprehensive Collector performs a much more thorough search than a Standard collector, collecting network data, processes, hashes, etc. We choose the specific data we wish to collect and then export it to an external flash drive. Then, we run the collector on our target machine through a provided shell script, extract the required data, and then bring it back to the original system for analysis. One crucial feature of Redline is its ability to read a filesystem and look through it in a convenient manner.

Other features exist in Redline, but they are simply GUI versions of ones already in Volatility, such as looking through processes, network analysis, process trees, etc.

4 Volatility Profiles

In this we see how to analyse memory dumps from newer editions of Windows, specifically Windows 10 and its Creator's and Fall Creator's updates. An example of a memory dump from the Windows build 16299 is given. This was not available even on the rolling-release versions of Linux at the time, so we directly clone the git repository to access the latest version of Volatility. Before that, we scanned the dump using selected versions of Windows, some of which gave us gibberish but nothing more.

5 Windows Process Genealogy

This is an in-depth look at the Windows process tree.

- **System:** This is the root of all Windows processes. Its user is **boot** and it has no executable from which it runs.
- **smss.exe:** Our session manager, associated with the account **local-system**. Its first instance spawns several child instances for each session.
  - **csrss.exe:** The client-server runtime. Created by an instance of **smss.exe** that exits instantly after creating it. Used for implementing much of the Windows API, among other functions.
  - **wininit.exe:** Starts key background processes, such as **services.exe** and **lsass.exe**.
    - **services.exe:** Manages all the services we find at boot time.
      - **svchost.exe:** Very commonly used by malware authors, often misspelled or running under incorrect parent processes if malicious. Can have multiple instances, usually ranging from 5-10.
        1. **taskhostw.exe:** Runs all Windows scheduled tasks, typically in an infinite loop listening for a "trigger" event. Renamed from "taskhost" in Windows 10.
        2. **runtimebroker.exe:** Facilitates interaction between UWP (Universal Windows Platform) apps and the Windows API. Also introduced in Windows 10.
    - **lsaiso.exe:** Only present when **Credential Guard** is. Secures secrets to only be accessible to select software. Introduced in Windows 10.
    - **lsass.exe:** Local security subsystems service. A critical process, meaning if it crashes the entire system does.
  - **winlogon.exe:** Handles user log-ins and log-offs, and runs the logon UI.
    - **userinit.exe:** Starts all the processes that occur when a user logs on.
      - **explorer.exe:** Gives us the interactive GUI and access to our filesystem. Has one instance per logged-on user.

6 Pulling Threads

The video simply walks through the process of memory analysis step by step. Starting it is the same process of cloning the Volatility git repository and finding our required profile.

Then we begin with running the command `malprocfind` on the memory file. We look for **False** values across multiple columns in the output, since it indicates malware. Pid 2888 is interesting, as it fits this criteria as well as has an incorrect PPid, pointing to `cmd.exe` instead of `services.exe`.

Then, executing the command `cmdline` and finding the one command after the process of 2888, we find `svchost.exe` being run with a suspicious `-K` flag.

We then use `netscan`, and then use the shell utility `grep` out any instances of default port connections such as on 80 or 443. This leads us to discover a connection with `powershell.exe`, which is extremely out of the ordinary and warrants further investigation.

After that, we dump the process executable and memory of the one with Pid 2888 and 3456, which we can use to perform further analysis on, such as hashing them and running the hashes through VirusTotal.

7 Persistence

This deals with ESAPs, which stand for Windows Autostart Extensibility Points. They are a common attack vector for hackers to make sure their code can run on reboots. A new Volatility plugin is introduced, called **winesap**, which is a modernized version of the Volatility **autoruns** plugin. A few methods are discussed to use this plugin, as the source code uses the same class name **AutoRuns**.

8 Baselines

In it, we compare two memory files of the same system, one before infection and one after. We find three Pids of importance, of which we dump the binaries of. Hashing these three executables, we find two of them are indeed malware from VirusTotal checks.

9 Prefetch

This section deals with Prefetch files and the Volatility plugins used in analyzing them. A Prefetch file contains information about the files loaded by the executable such as libraries and DLLs, and is used to optimize its execution. This comes of use because we can see multiple execution times of a single executable, and is thus highly valuable. Before that, we also find out the algorithm used by Windows in compressing the prefetch files— the Xpress Huffman algorithm. Only through downloading this can we read the files and extract valuable data from them.