DFRWS EU 2025 Workshop
Practical Malware Analysis and Memory Forensics for Incident Response
Faculty of Information Technology, Brno University of Technology (Czech Republic)

Memory Dump Analysis with Volatility 3

In this lab, you will learn how to analyze memory dumps as part of the malware analysis process, using the Volatility framework. We will work specifically with Volatility version 3 to examine a memory dump available on the workshop webpage[1]. The dump was obtained from a Windows 7 computer infected with the WannaCry ransomware, providing a real-world scenario for memory forensics. The sample has the following MD5 hash:`84c82835a5d21bbcf75a61706d8ab549`.

1 Volatility 2 vs. Volatility 3

While Volatility 2 has long been the standard in memory forensics, Volatility 3 represents a complete rewrite with several important changes. The most notable difference is Volatility 3's modular plugin architecture, which allows for better portability across operating systems and facilitates support for new file formats and memory layouts. Volatility 3 no longer requires specifying a *profile*, as it automatically detects system characteristics during analysis.

Command syntax has also changed: plugin names now follow a namespace structure (e.g., `windows.pslist.PsList` instead of just `pslist`). Additionally, Volatility 3 prioritizes structured output formats such as JSON, making it easier to integrate with automation tools or analysis pipelines. However, it's worth noting that not all Volatility 2 plugins have been ported yet, and some complex features like malware detection heuristics may be more mature in Volatility 2. As a result, forensic analysts may choose one or the other depending on the case requirements and plugin availability.

☞ Summary (TL;DR)

- Volatility 3 is a complete redesign of the framework with a modular architecture and improved portability.
- It no longer requires profile selection and automatically detects system characteristics.
- Plugin names use a namespace format (e.g., `windows.pslist.PsList`) instead of flat names.
- Volatility 3 supports structured output (such as JSON), allowing for integration with other tools and automation.
- Some Volatility 2 plugins are not yet available in Volatility 3, so you can continue to use both plugins as needed.

[1] See https://webdiis.unizar.es/~ricardo/dfrws-eu-25-workshop

2 Identifying the Source Machine

Before delving deeper into the analysis, it is often helpful to understand the basic characteristics of the system from which the memory dump was obtained. In Volatility 3, the `windows.info.Info` plugin provides details about the operating system, kernel version, service pack, architecture, and more. This information helps confirm the memory profile and help you choose the correct plugins or interpret artifacts accurately. Run the following command:

```
vol3 -f /shared/wannacry.elf windows.info.Info
```

The output will include values such as kernel base, KDBG, major/minor version, and service pack, confirming that this memory dump was obtained from a Windows 7 system. These details also provide useful context when correlating artifacts such as registry structures, process naming conventions, and DLL paths.

☞ Summary (TL;DR)

- The `windows.info.Info` plugin provides system-level information, including the operating system version, architecture, and kernel base.
- This helps confirm the memory profile and guides plugin selection during analysis (specially in Volatility 2).
- In this case, the memory dump originated from a Windows 7 system, which influences the interpretation of artifacts (e.g., registry layout or process tree structure).

3 Process Identification

The first step in memory forensics is to identify the processes active on the system at the time the memory was captured. This provides an overview of system activity and can help identify potential malicious behavior. Using Volatility 3's `PsList` and `PsTree` plugins, we can extract detailed information about each process, including its name, process ID (PID), parent process ID (PPID), number of threads and handles, session ID, creation time, and whether it was a 32-bit process running on a 64-bit operating system (WoW64).

To identify suspicious processes, there are several key elements to focus on. One of the most obvious red flags is the process name. While legitimate processes have well-known and consistent names, malware often uses unusual, random, or closely mimic names of legitimate system processes to avoid detection (e.g., `svch0st.exe` instead of `svchost.exe`). In our case, the `@WanaDecryptor` and `ed01ebfbc9eb5b` processes clearly stand out due to their unusual and suspicious names (see Figure 1).

Another important factor is the parent-child relationship between the processes. Legitimate Windows processes typically follow a predictable hierarchy. For example, `smss.exe` is usually the first user-mode process, followed by `csrss.exe` and `wininit.exe`, and so on. Any deviation from this hierarchy, such as a user application spawning unknown or strangely named child processes, should raise suspicion. Using the `PsTree` plugin, we observed that the suspicious `@WanaDecryptor` processes are children of `ed01ebfbc9eb5b`, spawned by `explorer.exe`, suggesting that the malware was executed during a logged-in user session (see Figure 2).

The process creation time is also an important clue. Processes created shortly before the memory snapshot, especially if they were not part of the normal startup sequence, could be


```
➜  ~ vol3 -f /shared/wannacry.elf windows.pslist.PsList
Volatility 3 Framework 2.25.0
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        File output

4       0       System  0x841389c8      85      496     N/A     False   2021-07-15 04:13:26.000000 UTC  N/A     Disabled
240     4       smss.exe        0x8553dd20      2       29      N/A     False   2021-07-15 04:13:26.000000 UTC  N/A     Disabled
320     312     csrss.exe       0x85853a58      8       657     0       False   2021-07-15 04:13:28.000000 UTC  N/A     Disabled
356     312     wininit.exe     0x85303d20      3       76      0       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
364     348     csrss.exe       0x85303a10      7       269     1       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
404     348     winlogon.exe    0x8508f5e8      5       121     1       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
448     356     services.exe    0x857ea7f0      9       254     0       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
456     356     lsass.exe       0x858111e0      8       796     0       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
464     356     lsm.exe 0x858cc030      11      207     0       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
568     448     svchost.exe     0x858db8b0      16      375     0       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
632     448     VBoxService.ex  0x8599a9e0      12      116     0       False   2021-07-15 04:13:29.000000 UTC  N/A     Disabled
```

Figure 1: Output of `PsList` for the WannaCry-infected memory dump.

```
832     840     explorer.exe    0x8597bb68      52      1262    1       False   2021-07-14 19:13:40.000000 UTC  N/A     \Device\HarddiskVolume2\Windows\explorer.exe    C:\Windows\Exp
lorer.EXE       C:\Windows\Explorer.EXE
* 3812  832     ed01ebfbc9eb5b  0x858282a8      10      83      1       False   2021-07-14 19:21:30.000000 UTC  N/A     \Device\HarddiskVolume2\Users\IEUser\Desktop\ed01ebfbc9eb5bbea
545af4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin\ed01ebfbc9eb5bbea545af4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin (1).exe       "C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545a
f4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin\ed01ebfbc9eb5bbea545af4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin (1).exe"   C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f
10716618404804 39c6e5babe8e080e41aa.bin\ed01ebfbc9eb5bbea545af4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin (1).exe
** 816  3812    @WanaDecryptor  0x8502d030      2       60      1       False   2021-07-14 19:22:02.000000 UTC  N/A     \Device\HarddiskVolume2\Users\IEUser\Desktop\ed01ebfbc9eb5bbea
545af4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin\@WanaDecryptor@.exe @WanaDecryptor@.exe co  C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f10716618404804 39c6e5babe8e080e41
aa.bin\@WanaDecryptor@.exe
*** 3920       816     taskhsvc.exe    0x84bfa3a8      6       119     1       False   2021-07-14 19:22:05.000000 UTC  N/A     \Device\HarddiskVolume2\Users\IEUser\Desktop\ed01ebfbc
9eb5bbea545af4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin\TaskData\Tor\taskhsvc.exe   TaskData\Tor\taskhsvc.exe       C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f10716618
40480439c6e5babe8e080e41aa.bin\TaskData\Tor\taskhsvc.exe
** 1252 3812    @WanaDecryptor  0x84adf030      1       63      1       False   2021-07-14 19:22:18.000000 UTC  N/A     \Device\HarddiskVolume2\Users\IEUser\Desktop\ed01ebfbc9eb5bbea
545af4d01bf5f10716618404804 39c6e5babe8e080e41aa.bin\@WanaDecryptor@.exe @WanaDecryptor@.exe     C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f10716618404804 39c6e5babe8e080e41
aa.bin\@WanaDecryptor@.exe
```

Figure 2: Output of `PsTree` for the WannaCry-infected memory dump, showing the process hierarchy and parent-child relationships.

involved in recent or ongoing activity. Additionally, an unusually high or low number of threads or handles, or processes running with unexpected session IDs, can also indicate anomalous behavior. For example, malware running in a user session might attempt to integrate with legitimate applications by mimicking their behavior or names.

By correlating all of these elements (process names, parent-child relationships, creation times, and session context), we can gain a deeper understanding of process activity and identify those that require further investigation. In this memory dump, the process tree reveals a pattern consistent with WannaCry's behavior, including a renamed Tor component (`taskhsvc.exe`, trying to disguise the user) as a child of the ransomware process. WannaCry uses this process to disguise itself as the legitimate Windows Task Scheduler service (`taskschd.exe`), blending into the user's session to evade detection and appear as a benign system component. This structured approach forms the basis for further analysis in subsequent steps.

☞ Summary (TL;DR)

- Suspicious processes can be identified by unusual names, unexpected parent-child relationships, creation time, and session context.
- Plugins such as `PsList` and `PsTree` help visualize the process list and hierarchy.
- In the WannaCry sample, `@WanaDecryptor`, `ed01ebfbc9eb5b`, and `taskhsvc.exe` stand out as abnormal processes.
- Understanding process hierarchy and metadata is essential for identifying malware execution chains.


```
→  ~ vol3 -f /shared/wannacry.elf windows.dlllist.DllList --pid 3920
Volatility 3 Framework 2.25.0
Progress:  100.00               PDB scanning finished
PID     Process Base    Size    Name    Path    LoadTime        File output

3920    taskhsvc.exe    0x1090000       0x2fe000        taskhsvc.exe    C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840
480439c6e5babe8e080e41aa.bin\TaskData\Tor\taskhsvc.exe        N/A     Disabled
3920    taskhsvc.exe    0x770c0000      0x142000        ntdll.dll       C:\Windows\SYSTEM32\ntdll.dll   N/A     Disabled
3920    taskhsvc.exe    0x756b0000      0xd5000 kernel32.dll    C:\Windows\system32\kernel32.dll        2021-07-14 19:22:05.000000 UTC
Disabled
3920    taskhsvc.exe    0x75250000      0x4b000 KERNELBASE.dll  C:\Windows\system32\KERNELBASE.dll      2021-07-14 19:22:05.000000 UTC
Disabled
3920    taskhsvc.exe    0x62d20000      0x82000 libevent-2-0-5.dll      C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840
480439c6e5babe8e080e41aa.bin\TaskData\Tor\libevent-2-0-5.dll  2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x6b5d0000      0x1c000 libssp-0.dll    C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6
e5babe8e080e41aa.bin\TaskData\Tor\libssp-0.dll2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x75b20000      0xa1000 ADVAPI32.dll    C:\Windows\system32\ADVAPI32.dll        2021-07-14 19:22:05.000000 UTC
Disabled
3920    taskhsvc.exe    0x77220000      0xac000 msvcrt.dll      C:\Windows\system32\msvcrt.dll  2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x75b00000      0x19000 sechost.dll     C:\Windows\SYSTEM32\sechost.dll 2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x75a50000      0xa2000 RPCRT4.dll      C:\Windows\system32\RPCRT4.dll  2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x624e0000      0x77000 libgcc_s_sjlj-1.dll     C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840
480439c6e5babe8e080e41aa.bin\TaskData\Tor\libgcc_s_sjlj-1.dll 2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x75f00000      0xc4c000        SHELL32.dll     C:\Windows\system32\SHELL32.dll 2021-07-14 19:22:05.000000 UTC
Disabled
3920    taskhsvc.exe    0x76f90000      0x57000 SHLWAPI.dll     C:\Windows\system32\SHLWAPI.dll 2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x75590000      0x4e000 GDI32.dll       C:\Windows\system32\GDI32.dll   2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x755e0000      0xc9000 USER32.dll      C:\Windows\system32\USER32.dll  2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x77210000      0xa000  LPK.dll C:\Windows\system32\LPK.dll     2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x76cc0000      0x9d000 USP10.dll       C:\Windows\system32\USP10.dll   2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x772d0000      0x35000 WS2_32.dll      C:\Windows\system32\WS2_32.dll  2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x76f80000      0x6000  NSI.dll C:\Windows\system32\NSI.dll     2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x622c0000      0x21c000        LIBEAY32.dll    C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840
480439c6e5babe8e080e41aa.bin\TaskData\Tor\LIBEAY32.dll        2021-07-14 19:22:05.000000 UTC  Disabled
3920    taskhsvc.exe    0x62230000      0x82000 SSLEAY32.dll    C:\Users\IEUser\Desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6
e5babe8e080e41aa.bin\TaskData\Tor\SSLEAY32.dll2021-07-14 19:22:05.000000 UTC  Disabled
```

Figure 3: Output of `DllList` with PID 3920 for the WannaCry-infected memory dump.

4 DLL Analysis

Once a suspicious process has been identified, the next step is to analyze the dynamic link libraries (DLLs) it loaded during execution. Recall that DLLs are shared libraries that provide functionality such as file access, networking, user interface controls, or cryptographic operations. By inspecting the DLLs loaded into a process's memory, we can infer what capabilities it might be using and potentially uncover artifacts related to malicious behavior.

In Volatility 3, this can be done using the `DllList` plugin, which lists all modules loaded by a specific process. For instance, analyzing the `taskhsvc.exe` process (PID 3920, see Figure 3), reveals several Tor-related libraries, such as ≪`libevent-2-0-5.dll`≫, ≪`LIBEAY32.dll`≫, and ≪`SSLEAY32.dll`≫. These DLLs are uncommon in standard Windows installations and clearly point to third-party or custom functionality embedded in malware. The paths from which these DLLs are loaded also provide additional context: in this case, they are loaded from directories such as `C:\...\TaskData\Tor\`, confirming their relationship to the Tor network client included in WannaCry.

We also observed standard Windows cryptographic and network libraries, such as `WS2_32.dll`, `mswsock.dll`, and `dhcpcsvc.dll`, which are commonly used for socket communication and network configuration, supporting the hypothesis that the process handles outgoing communication, likely with command and control servers.

Similarly, the `@WanaDecryptor` process (PID 816) loads a different set of libraries, including those related to the Microsoft Visual C++ runtime (`MFC42.dll`, `MSVCRT.dll`), cryptographic services (`CRYPT32.dll`), registry manipulation (`ADVAPI32.dll`), and web access (`WININET.dll`, `urlmon.dll`). These DLLs suggest that the process interacts with the registry, performs encryption/decryption, and may present a graphical interface to the user. All of this is consistent with the well-known behavior of WannaCry ransomware.

By analyzing the loaded DLLs and their paths, we gain insight into each process's behavior, intentions, and potential role within the overall attack. Suspicious or uncommon DLLs, unusual load paths, and the presence of third-party libraries in non-standard locations are strong indica-

tors that a process is malicious or compromised. This step lays the groundwork for analysis and specific code extraction in the next phase.

```
vol3 -f /shared/wannacry.elf -r json windows.dlllist.DllList --pid 816 | grep Path
Volatility 3 Framework 2.25.0
    "Path": "C:\\Users\\IEUser\\Desktop\\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.bin\\@WanaDecryptor@.exe",
    "Path": "C:\\Windows\\SYSTEM32\\ntdll.dll",
    "Path": "C:\\Windows\\system32\\kernel32.dll",
    "Path": "C:\\Windows\\system32\\KERNELBASE.dll",
    "Path": "C:\\Windows\\system32\\MFC42.DLL",
    "Path": "C:\\Windows\\system32\\msvcrt.dll",
    "Path": "C:\\Windows\\system32\\USER32.dll",
    "Path": "C:\\Windows\\system32\\GDI32.dll",
    "Path": "C:\\Windows\\system32\\LPK.dll",
    "Path": "C:\\Windows\\system32\\USP10.dll",
    "Path": "C:\\Windows\\system32\\ole32.dll",
    "Path": "C:\\Windows\\system32\\RPCRT4.dll",
    "Path": "C:\\Windows\\system32\\OLEAUT32.dll",
    "Path": "C:\\Windows\\system32\\ODBC32.dll",
    "Path": "C:\\Windows\\system32\\ADVAPI32.dll",
    "Path": "C:\\Windows\\SYSTEM32\\sechost.dll",
    "Path": "C:\\Windows\\system32\\SHELL32.dll",
    "Path": "C:\\Windows\\system32\\SHLWAPI.dll",
    "Path": "C:\\Windows\\WinSxS\\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\\COMCTL32.dll",
    "Path": "C:\\Windows\\system32\\urlmon.dll",
    "Path": "C:\\Windows\\system32\\XmlLite.dll",
    "Path": "C:\\Windows\\system32\\WININET.dll",
    "Path": "C:\\Windows\\system32\\iertutil.dll",
    "Path": "C:\\Windows\\system32\\CRYPT32.dll",
    "Path": "C:\\Windows\\system32\\MSASN1.dll",
    "Path": "C:\\Windows\\system32\\MSVCP60.dll",
    "Path": "C:\\Windows\\system32\\WS2_32.dll",
    "Path": "C:\\Windows\\system32\\NSI.dll",
    "Path": "C:\\Windows\\system32\\IMM32.DLL",
    "Path": "C:\\Windows\\system32\\MSCTF.dll",
    "Path": "C:\\Windows\\system32\\odbcint.dll",
    "Path": "C:\\Windows\\system32\\RICHED32.DLL",
    "Path": "C:\\Windows\\system32\\RICHED20.dll",
    "Path": "C:\\Windows\\system32\\uxtheme.dll",
    "Path": "C:\\Windows\\system32\\dwmapi.dll",
    "Path": "C:\\Windows\\system32\\mswsock.dll",
    "Path": "C:\\Windows\\System32\\wshtcpip.dll",
    "Path": "C:\\Windows\\system32\\apphelp.dll",
```



```
➜  ~ vol3 -f /shared/wannacry.elf windows.handles --pid 3812 | grep Mutant
3812ressed01ebfbc9eb5b  0x85d22650B scan0x30 finMutant  0x1f0001        -
3812    ed01ebfbc9eb5b  0x855e59b0      0x34    Mutant  0x1f0001        -
3812    ed01ebfbc9eb5b  0x8561e9e8      0x6c    Mutant  0x1f0001        MsWinZonesCacheCounterMutexA
3812    ed01ebfbc9eb5b  0x85562278      0x74    Mutant  0x1f0001        MsWinZonesCacheCounterMutexA0
3812    ed01ebfbc9eb5b  0x852ab768      0x150   Mutant  0x1f0001        -
```

Figure 4: Output of `Handles` with PID 3812, filtered by "Mutant", for the WannaCry-infected memory dump.

☞ Summary (TL;DR)

- The `DllList` plugin shows all dynamic-link libraries loaded by a process, offering insight into its capabilities and external dependencies.
- Observing non-standard DLLs and unusual load paths can reveal embedded components or third-party functionality.
- DLL analysis supports behavioral profiling and complements static inspection of the executable.

5 Identifying Artifacts of Interest

After identifying suspicious processes and analyzing their associated DLLs, the next step is to look for specific artifacts that can confirm malicious behavior. These artifacts often include mutexes, named pipes, event handlers, or registry objects–resources that malware often uses to coordinate execution, maintain persistence, or prevent re-infection, respectively.

In the case of WannaCry, a particularly relevant artifact is the mutex object it creates at runtime. Recall that a mutex (short for *mutual exclusion*) is often used by malware to ensure that only one instance of a process is running. WannaCry creates a mutex with a unique name (`MsWinZonesCacheCounterMutexA`) as an infection marker. If the malware finds this mutex already present in memory, it assumes the system has already been infected and terminates its execution to avoid reinfecting the computer. This is a common technique among malware families to reduce noise, improve efficiency, and avoid detection by repeating the same behavior.

To identify such objects in memory, you can use the Volatility 3 `Handles` plugin in combination with `grep` to filter the output and only show handles of type Mutant (the Windows internal object type for mutexes). See Figure 4 for an example.

☞ Summary (TL;DR)

- Malware often creates runtime objects such as mutexes to prevent reinfection or coordinate execution.
- The `Handles` plugin can reveal these artifacts, especially when filtered by type (e.g., `Mutant` for mutexes).
- In the WannaCry case, the presence of `MsWinZonesCacheCounterMutexA` serves as an infection marker.
- These runtime indicators are useful for detection and for verifying malware behavior even if names or binaries are obfuscated.

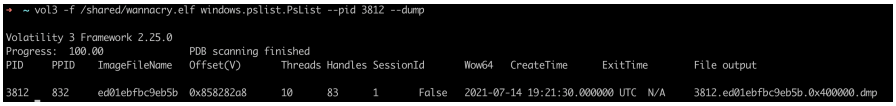

Figure 5: Output of `PsList` with PID 3812, showing the full process dumped to file from the WannaCry-infected memory dump.

6 Dumping Processes and DLLs

Once suspicious processes and associated libraries have been identified, a common next step is to extract them from the memory dump for offline analysis. This allows us to reverse engineer the binaries using tools such as disassemblers, debuggers, or sandbox environments, without needing to interact directly with the live infected system.

To dump the memory of an entire process, you can use the `PsList` plugin with the `--dump` flag. For example, to extract the process with PID 3812 (identified earlier as the core WannaCry component), run the following command:

```
vol3 -f /shared/wannacry.elf windows.pslist.PsList --pid 3812 --dump
```

An output example is shown in Figure 5. This will generate a file (e.g., `pid.3812.0x400000.dmp`) containing the in-memory image of the process, which can then be loaded into tools like Ghidra, IDA Pro, or PEStudio for static or dynamic analysis. This enables a deeper understanding of the malware's behavior, such as how encryption routines are implemented or how persistence mechanisms are triggered.

In addition to dumping the full process, it is often valuable to extract individual DLLs loaded by the process–especially if they appear suspicious or were loaded from unusual paths. To do this, use the `DllList` plugin with the `--dump` flag. For instance:

```
vol3 -f /shared/wannacry.elf windows.dlllist.DllList --pid 3812 --dump
```

Each DLL will be extracted into its own file, named using the module name and base address, allowing for precise identification (see an output example in Figure 6). This is particularly useful for isolating custom or third-party DLLs, such as those mentioned earlier, e.g., ≪`SSLEAY32.dll`≫ or ≪`LIBEAY32.dll`≫, which were bundled with WannaCry's embedded Tor client.

Extracted binaries –both executables and DLLs– can then be analyzed for strings, function imports, embedded resources, and behavioral patterns. This step bridges the gap between memory forensics and traditional malware reverse engineering, allowing analysts to confirm the purpose of suspicious components and potentially uncover new indicators of compromise.

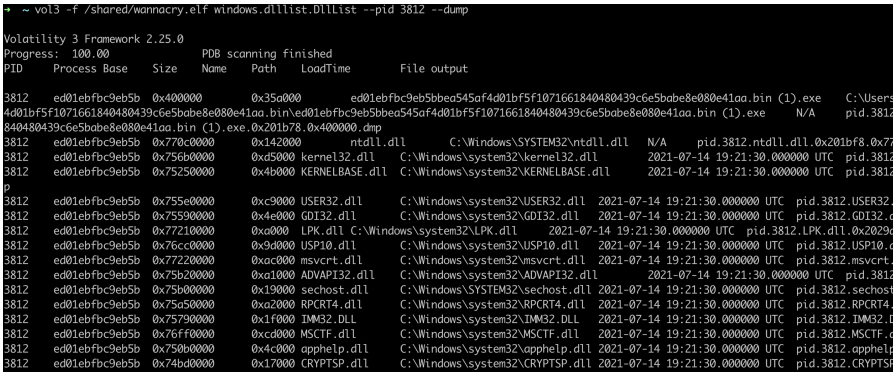

Figure 6: Output of `DllList` with PID 3812, showing each loaded DLL dumped to a separate file from the WannaCry-infected memory dump.

☞ Summary (TL;DR)

- Extracting binaries from memory allows for detailed offline analysis using reverse engineering tools like Ghidra or IDA Pro.
- The `--dump` option in plugins like `PsList` and `DllList` enables full process or library dumping.
- Dumped binaries can be subjected to string analysis, static inspection, or sandbox execution.
- This step bridges memory forensics with traditional malware reverse engineering (also known as *malware analysis*).

7 Persistence Identification

One of the key objectives of any malware is to maintain persistence on the infected system, ensuring it continues running even after a reboot or logout. In incident response, identifying persistence mechanisms is required to fully understand the infection and how the threat maintains control of the system, recommending effective remediation measures (e.g., to ensure that the system is not reinfected upon reboot, even after the malicious process has terminated).

On Windows systems, persistence is commonly achieved through various Autostart Extensibility Points (ASEPs), such as registry keys, services, scheduled tasks, startup folders, or DLL load points. WannaCry, like many ransomware variants, attempts to establish persistence by modifying specific registry keys within the system. Analysts should prioritize investigating the `Run`, `RunOnce`, and `Services` keys, as malware frequently uses them to ensure execution upon startup on Windows systems.

To examine these keys, we can use the Volatility 3 `PrintKey` plugin, which extracts the contents of the Windows registry in memory:

```
vol3 -f /shared/wannacry.elf windows.registry.printkey.PrintKey
```

This plugin provides access to registry hives such as `HKEY_LOCAL_MACHINE_SYSTEM` and

`HKEY_CURRENT_USER`, including keys such as `ControlSet001` or `Run` entries, which malware often uses to trigger execution at system startup. By reviewing these keys and their associated values, we can detect entries pointing to suspicious binaries or unknown executables.

In this case, although WannaCry is primarily known for its rapid spread and encryption, it can also include components designed to restart the ransomware after a reboot. These are typically located in paths such as the user's desktop or temporary directories and registered using startup registry keys.

For more information on persistence mechanisms and how they appear in memory, you are encouraged to refer to the article "*Characteristics and Detectability of Windows Autostart Extensibility Points in Memory Forensics*" (DOI: 10.1016/j.diin.2019.01.026) [UR19]. This paper provides a systematic classification of Windows ASEPs and their detectability in forensic analysis.

☞ Summary (TL;DR)

- Malware often modifies the Windows Registry to establish persistence on reboot.
- The `PrintKey` plugin extracts registry keys from memory, including common persistence locations like `Run` keys.
- Analyzing registry entries can reveal startup mechanisms and additional payload paths.
- Understanding persistence is key to both eradication and incident documentation.

8 What's Next?

After completing this lab, you'll have a practical understanding of how to investigate a real malware infection using memory forensics with Volatility 3. You've learned how to identify suspicious processes, analyze their loaded DLLs, locate runtime artifacts like mutexes, and extract binaries for further analysis.

The next natural step is to delve deeper into *reverse engineering*. You can use tools like Ghidra or IDA Pro to examine the behavior of dumped binaries, analyze how they perform encryption, or understand how persistence is implemented (see, for instance, Figure 7). These tools allow you to inspect malware logic at the assembly level, providing a more detailed view than memory forensics alone.

We also encourage you to explore Volatility 3's more advanced features. Plugins like cmdscan, yarascan, or netscan (from Volatility 2) can provide information about command history, malware signatures in memory, or network activity. Comparing Volatility 2 and 3 results on the same memory dump can also help you appreciate the differences between the two versions and help you choose the right tool.

As you become more familiar with these tools, you may want to automate parts of your analysis. Writing simple scripts to calculate hashes, extract strings, or scan memory with YARA rules can save time and make your workflow more efficient. These automations are especially useful when analyzing multiple memory dumps or responding to large-scale incidents.

Finally, consider connecting the forensic artifacts you observe with frameworks like MITRE ATT&CK[2]. The MITRE ATT&CK framework is a structured knowledge base covering real-world adversary tactics and techniques used in cyberattacks. It classifies how attackers operate, from initial access to data exfiltration, providing a common language for describing threat behavior.

[2]See `https://attack.mitre.org/`.

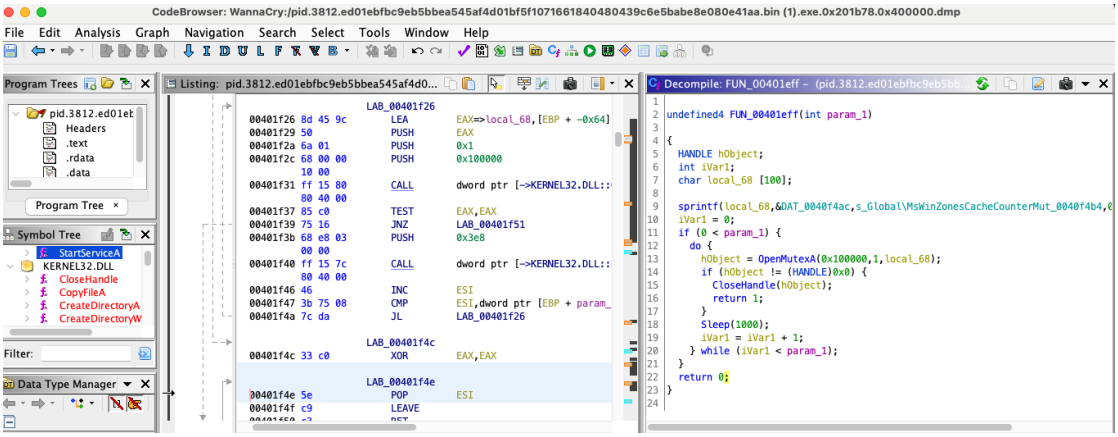

Figure 7: Analysis of the malicious sample in `Ghidra` 10.01, showing the code responsible for creating the mutex used by the malware (sample `84c82835a5d21bbcf75a61706d8ab549`).

Widely used in threat detection, incident response, and security engineering, ATT&CK helps analysts relate observed indicators (such as process creation, persistence mechanisms, or network activity) to known attack patterns, enabling more informed and effective defenses.

This will help you think more systematically about how threats behave and how to detect and mitigate them. Documenting your findings in a structured report is also an essential skill, especially in professional environments where communicating results is as important as technical accuracy.

This lab is just the beginning. Continued practice with real-world examples, combined with deeper study of malware techniques and forensic tools, will prepare you for more advanced investigations.

References

[UR19] Daniel Uroz and Ricardo J. Rodríguez. Characteristics and Detectability of Windows Auto-Start Extensibility Points in Memory Forensics. *Digital Investigation*, 28:S95–S104, April 2019.

