An Overview of Volatility

Jeff Malavasi
Dept. of Computing Security
Rochester Institute of Technology
Email: jm3378@rit.edu

I. Introduction

Memory analysis is a process in which a researcher captures a dump of volatile system memory in order to look for signs of malicious activity. This can prove to be extremely helpful in both forensic analysis of a system as well as classifying a novel binary as malicious or benign [1]. One drawback to memory analysis is that it produces an extremely large, noisy dataset that can be difficult to parse through. Volatility Memory Analysis is an open-source toolkit that is designed to quickly look for important signals within a memory dump. It can also be automatically integrated into sandbox technologies, such as Cuckoo Sandbox, in order to augment its capabilities. In this report, a Windows XP memory dump was provided for analysis. Additionally, Cuckoo Sandbox was used to analysis and capture a memory dump of an unknown malicious file.

II. Methods

A. Installation & Setup

Volatility can be installed as a standalone application on Windows, Linux, and Mac by downloading the latest release from GitHub [2]. After Volatility is installed, it can be pointed at any memory dump on the file system using a set of command line arguments. The first step is to determine the type of image captured in order to provide Volatility with a profile that can be used for further analysis. To do this, we must provide volatility with the parameter: **imageinfo**. In the below example, we can see the image is most likely a 32 bit version of Windows XP.

```
csec759@ubuntu:~/Desktop/Memory Images$ volatility -f "Windows XP SP3 x86.img" imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/csec759/Desktop/Memory Images/Windows XP SP3 x86.img)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80545b60L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2008-11-26 07:46:02 UTC+0000
     Image local date and time : 2008-11-26 02:46:02 -0500
csec759@ubuntu:~/Desktop/Memory Images$
```

B. Command Line

There are two useful plugins that can be used in order to gather information about which commands have been launched on the system. The first is the **commandline** plugin, which outputs all of the command line binaries and arguments called recently and groups the results by each process. Since the output includes all processes running at the time the analysis was captured, it can be difficult to interpret. One way that we can improve the output of this plugin is to provide a process ID or offset value. An offset value references the location in memory where the process originates. One way to determine if there are any suspicious processes is to run the **psxview** plugin, which will highlight any processes that may be evading detection. In the below example, we can see a suspicious process called *network_listener*, which is evading pslist and pssscan.

```
csec759@ubuntu:~/Desktop/Memory Images$ volatility -f "Windows XP SP3 x86.img" profile=WinXPSP3x86 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x01a2b100 winlogon.exe            620 True   True   True     True   True  True    True
0x01a3d360 svchost.exe             932 True   True   True     True   True  True    True
0x018a13c0 VMwareService.e        1756 True   True   True     True   True  True    True
0x018e75e8 spoolsv.exe            1648 True   True   True     True   True  True    True
0x019dbc30 lsass.exe               684 True   True   True     True   True  True    True
0x0184e3a8 wscntfy.exe             560 True   True   True     True   True  False   True
0x018af860 VMwareTray.exe         1896 True   True   True     True   True  True    True
0x01a4bc20 network_listene        1696 False  False  True     True   True  False   True
```

If we run the **commandline** plugin now against the offset 0x01a4bc20, we can see the location of the malicious file and potentially any arguments that were passed to it during execution. This could be useful if there were multiple code paths in the binary, and the malware was passing runtime parameters to potentially evade detection or be more robust against diverse operating systems.

```
csec759@ubuntu:~/Desktop/Memory Images$ volatility -f "Windows XP SP3 x86.img" profile=WinXPSP3x86 cmdline --offset=0x01a4bc20
Volatility Foundation Volatility Framework 2.6
************************************************************************
network_listene pid:   1696
Command line : "C:\Documents and Settings\moyix\Desktop\network_listener.exe"
csec759@ubuntu:~/Desktop/Memory Images$
```

While **commandline** can be used to show information about active processes, the second plugin **cmdscan** can be used to output bash history. While most commonly used for enumeration of secrets, this can also be useful in binary analysis.

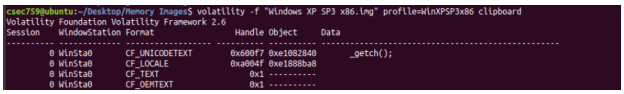
C. Privs

The next plugin, **privs**, can be used to show which privileges a given process currently has as well as which ones it can escalate if needed. This is useful because it can help determine what class of malware the binary belongs to. For example, a ransomware sample would likely need access to modify the file system in order to be effective. When analyzing the suspicious process from the Windows XP machine, we can see that it has the ability to modify the file system, change the state of the system, load device drives and monitors process and system performance.

```
csec759@ubuntu:~/Desktop/Memory Images$ volatility -f "Windows XP SP3 x86.img" profile=WinXPSP3x86 privs --offset=0x01a4bc20
Volatility Foundation Volatility Framework 2.6
Pid      Process              Value  Privilege                         Attributes                  Description
-------- -------------------- ------ --------------------------------- --------------------------- -----------
    1696 network_listene      23 SeChangeNotifyPrivilege           Present,Enabled,Default  Receive notifications of changes to files or directories
    1696 network_listene       8 SeSecurityPrivilege               Present                  Manage auditing and security log
    1696 network_listene      17 SeBackupPrivilege                 Present                  Backup files and directories
    1696 network_listene      18 SeRestorePrivilege                Present                  Restore files and directories
    1696 network_listene      12 SeSystemtimePrivilege             Present                  Change the system time
    1696 network_listene      19 SeShutdownPrivilege               Present                  Shut down the system
    1696 network_listene      24 SeRemoteShutdownPrivilege         Present                  Force shutdown from a remote system
    1696 network_listene       9 SeTakeOwnershipPrivilege          Present                  Take ownership of files/objects
    1696 network_listene      20 SeDebugPrivilege                  Present                  Debug programs
    1696 network_listene      22 SeSystemEnvironmentPrivilege      Present                  Edit firmware environment values
    1696 network_listene      11 SeSystemProfilePrivilege          Present                  Profile system performance
    1696 network_listene      13 SeProfileSingleProcessPrivilege   Present                  Profile a single process
    1696 network_listene      14 SeIncreaseBasePriorityPrivilege   Present                  Increase scheduling priority
    1696 network_listene      10 SeLoadDriverPrivilege             Present,Enabled          Load and unload device drivers
    1696 network_listene      15 SeCreatePagefilePrivilege         Present                  Create a pagefile
    1696 network_listene       5 SeIncreaseQuotaPrivilege          Present                  Increase quotas
    1696 network_listene      25 SeUndockPrivilege                 Present,Enabled          Remove computer from docking station
    1696 network_listene      28 SeManageVolumePrivilege           Present                  Manage the files on a volume
    1696 network_listene      29 SeImpersonatePrivilege            Present,Enabled,Default  Impersonate a client after authentication
    1696 network_listene      30 SeCreateGlobalPrivilege           Present,Enabled,Default  Create global objects
csec759@ubuntu:~/Desktop/Memory Images$
```

D. Clipboard

The final plugin that was explored is called **clipboard**. This plugin is able to extract the contents of the clipboard from all uses on the system. This can be useful when enumerating for sensitive information such as passwords or API keys. With the proliferation of password managers, many sensitive tokens sit on the clipboard of a machine for long periods of time. In the below example, volatility was able to extract the unicode string *_getch()*. This string is a commonly used C function that reads input from the keyboard. This may indicate the presence of a keylogger, which is a type of malware used to capture and send all keystrokes from a victim machine to an attacker.



III. Conclusion

Volatility memory analysis can be used to quickly and effectively parse through a memory dump. Through the use of both native and third-party plugins, many different behavioral features can be extracted. In the initial sample, we were able to uncover a malicious process called that had a high number of API privileges. Additionally, on the sample created with Cuckoo sandbox, many processes spawned by Cuckoo itself are hidden, which helps to evade detection by malware. While only a few plugins have been highlighted in this report, there are many other ones that can be furthered used to analyze the memory dumps.

References

[1] H. Macht, "Live memory forensics on android with volatility," *Friedrich-Alexander University Erlangen-Nuremberg*, 2013.

[2] V. Foundation, "Volatility docs," https://github.com/volatilityfoundation/volatility/wiki, 2023.