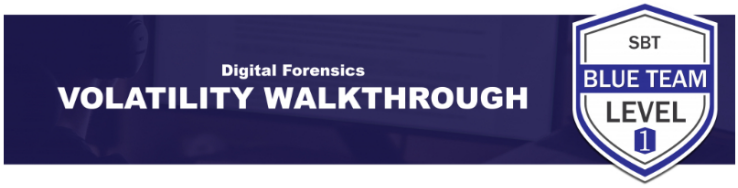
Volatility Walkthrough



In this video, we will show you how to analyze a memory dump using some basic plugins within Volatility. Before we jump into that, you need to understand a fundamental concept about how Volatility actually works:

- **Volatility needs profiles to work.** When we have the memory image file we want to analyze we first need to use the command **`volatility -f memdump.mem imageinfo`**. Once this command is run, Volatility will identify the system the memory image was taken from, including the operating system, version, and architecture. For example, if we took a memory image from a Windows 7 machine with Service Pack 1 and it had a 64-bit architecture, Volatility would tell us the best profile to use is Win7SP1x64. In the below screenshot, we can see that this memory image has been given a suggested profile of WinXPSP2x86 (Windows XP, Service Pack 2, 32-bit architecture). When running any other command on this memory image we need to provide the profile somewhere, in the format **`--profile=WinXPSP2x86`**, otherwise, the command will not run.

```
root@SBTLab2:~/Desktop# volatility -f cridex.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG searc
h ...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated
 with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/root/Desktop/cridex
.vmem)
                      PAE type : PAE
                           DTB : 0x2fe000L
                          KDBG : 0x80545ae0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2012-07-22 02:45:08 UTC+0000
     Image local date and time : 2012-07-21 22:45:08 -0400
```

Now that you understand how to retrieve the suggested profile and use it in future commands, watch this video where we perform some basic analysis of a memory dump. Below the video is a transcript, and below that is a command list to help you with the exercise. Take notes!

Transcript

> *In this video we're going to show you how to analyze memory using Volatility.*
>
> *For this example, we're going to analyze memdump1.mem. The first thing we need to do is*

run the image info command to discover what profile we need to use for analysis.

Use the command `volatility -f memdump1.mem imageinfo`. *Once the command finishes, we can see the suggested profile, Win7SP1x64, meaning Windows 7, Service Pack 1, with a 64-bit architecture. We can also see the KDBG address, and how many processors that system has, as well as the service pack.*

Now if we try to run a plugin, such as pslist to list processes, it wont work. We need to go back and include the profile in every command we use going forward. Let's run the command again, and we can now see lots of processes that were running on the system, including process IDs and timestamps.

The next command we're going to look at is pstree. This is the same as pslist, but displays it in a tree format instead.

Thirdly we'll use psscan. This plugin can identify hidden processes, often used by malware. During an investigation we would compare the results of pslist and psscan for differences.

Next is psxview, which is a combination of pslist and psscan, giving us lots of information about the processes.

Now lets take a look at network connection. Using the netscan plugin we can view active and closed network connections from the time of memory capture.

Another useful plugin is timeliner, which uses timestamps from activity within the memory dump in time order. This can be extremely useful for incident response, allowing responders to build a linear timeline of events.

Another command is iehistory, be careful, all plugins are case sensitive! This can allow us to view browsing history, in this case we can see the user visited msn.co.uk, likely because it is the default homepage for Microsoft edge.

The next command is filescan, which will list every single file mentioned in the memory dump. Note this list can be extremely long. Here we can see the system has FTKImager, and wireshark installed. This could be useful for identifying running programs and important files.

We can use the dumpfiles plugin to retrieve these files from the captured memory! We just need to select a dump location, and volatility will start retrieving every single file it can. We can see these files are now on our desktop.

You'll get a chance to analyze memory images yourself in the next lesson.

Command List

`volatility -f memdump.mem imageinfo` // Take memory image "memdump.mem" and determine the suggested profile for analysis. The profile is the operating system, version, and architecture.

`volatility -f memdump.mem --profile=PROFILE pslist` // Take memory image, provide the profile, then use the **pslist** plugin to print a list of processes to the terminal.

`volatility -f memdump.mem --profile=PROFILE pstree` // Use the **pstree** plugin to print a process tree to the terminal.

`volatility -f memdump.mem --profile=PROFILE psscan` // Use the **psscan** plugin to print all available processes, including hidden ones often used by malware (compare this to pslist to see if there's any differences!).

`volatility -f memdump.mem --profile=PROFILE psxview` // Use the plugin psxview plugin to print expected and hidden processes. This is a combination of pslist and psscan plugins.

`volatility -f memdump.mem --profile=PROFILE netscan` // Use the plugin **netscan** to identify any active or closed network connections.

`volatility -f memdump.mem --profile=PROFILE timeliner` // Use the **timeliner** plugin to create a timeline of events from the memory image.

`volatility -f memdump.mem --profile=PROFILE iehistory` // Use the **iehistory** plugin to pull internet browsing history.

`volatility -f memdump.mem --profile=PROFILE filescan` // Use the **filescan** plugin to identify any files on the system from the memory image.

`volatility -f memdump.mem --profile=PROFILE dumpfiles -n --dump-dir=./` // Use the **dumpfiles** plugin to retrieve files from the memory image. In this case our terminal is open in the Desktop (root@SBTLab2:~/Desktop) and we are using the output location `./` which tells Volatility to put the files in our current location, the Desktop.

Additional Resources

If you want to get some practice in before the exercise, or if you've finished the exercise and want to play around with Volatility some more, you are able to download open-source memory dumps at Volatility's own GitHub link, all created for analysis with Volatility. We strongly suggest that students try at least a few of these dumps to see if they can find anything interesting and become more confident using this tool.
https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples

You can find a great list of useful Volatility commands here:
https://book.hacktricks.xyz/forensics/volatility-examples#list-processes

We also recommend you read this article which gives a great insight on how to approach a memory investigation:
First steps to volatile memory analysis | by P4N4Rd1 | Medium

Quizzes

Lab) Memory Analysis Investigation

Previous Topic

Back to Lesson

Privacy & Cookies Policy