

Memory analysis workshop

Lab instructions only

Version 1.1 – updated 20240301

© Crown Copyright, The National Cyber Security Centre 2023. This information is licensed under the Open Government Licence v3.0. To view this licence, visit <https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/>.

When you use this information under the Open Government Licence, you should include the following attribution: CyBOK memory analysis workshop slides © Crown Copyright, The National Cyber Security Centre 2023, licensed under the Open Government Licence: <https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/>.

# Memory analysis workshop

This workshop introduces memory analysis for the purposes of digital forensics and malware analysis. It relates to selected topics of Cyber Security Body of Knowledge version 1.1 (CyBOK) knowledge areas “Forensics”, “Security Operations & Incident Management”, and “Malware and Attack Technologies”[[1]](#footnote-1). Other sources are, whenever used, cited throughout this document.

The workshop intends to introduce memory analysis and related necessary topics such as data storage in memory and properties of memory-resident data. The workshop will also describe when memory analysis can provide valuable results for forensic investigations and incident response processes. The workshop begins with an introduction to the theoretical concepts and continues with a hands-on lab.

This notebook will support the workshop. The first section contains study and reflection support for the theoretical components. The second section contains instructions for a hands-on lab. The third section provides a summary of the central concepts.

*Note! This document only contains the lab instructions for the lab associated with the workshop. The full workshop is available at* [*https://github.com/kavrestad/MalwareAnalysis*](https://github.com/kavrestad/MalwareAnalysis)

# Section 2: Hands-on Lab

This lab intends to demonstrate the basic functionality of Volatility memory analysis and introduces memory analysis techniques. Remember that malware analysis is, to a large extent, about finding abnormal behavior. While we can know a fair bit about what malware may typically try to do, it is hard to have a concrete guide. Rather, we need to understand what normal computer operations are and try to find deviations from that – detective work!

You will need the file MalwareSample.7z which you can download from: <https://github.com/kavrestad/MalwareAnalysis>

**NOTE! This is a memory sample containing a malware sample that infected Windows XP computers. While it should be harmless on modern operating systems, avoid doing this lab in sensitive environments!**

## Setting up the lab environment

Volatility 2.6 or Volatility 3 will be used for this lab. Volatility 3 is newer and Volatility 2.6 is discontinued. However, there are differences between the two tools that merit including both of them in the toolbox for a forensic analyst. The most obvious difference is that Volatility 2.6 is discontinued and may therefore not work well for the analysis of modern operating systems. However, Volatility 2.6 is still more extensive in terms of functionality than Volatility 3.

Volatility is a command-line tool built on Python. The Lab instructions contain one track for each version of Volatility. Volatility 2.6 is included first because it is easier to install. Note that the best installation alternative is to always pull the latest version from GitHub. However, the simpler installation procedures explained below are sufficient for this Workshop. This lab is written for Windows, but Volatility can be used similarly on Linux.

### Volatility 2.6 installation in Windows

* Go to <https://github.com/kavrestad/MalwareAnalysis>
* Download Volatility 2.6 standalone executable
* Open a PowerShell terminal and navigate to the folder containing Volatility

### Volatility 3 installation in Windows

*Note that you need to have python 3.6 or later installed*

* Download AND UNPACK Volatility 3
  + From the web page: <https://www.volatilityfoundation.org/3>
  + Clone from git (ensures the latest version): git clone <https://github.com/volatilityfoundation/volatility3.git>
* Install some dependencies
  + *Python -m pip install pefiles*
* Open a PowerShell terminal and navigate to the Volatility folder containing “vol.py”
* You can now run Volatility by issuing the command *python vol.py*

## Testing Volatility

In this section, you will experience the basic usage of Volatility. We will begin with volatility 2.6 and then do the same things with volatility 3. Before we start, download and unpack the file called MalwareSample. The archive contains a memory sample infected with the malware Cridex.

### Volatility 2.6 basic usage

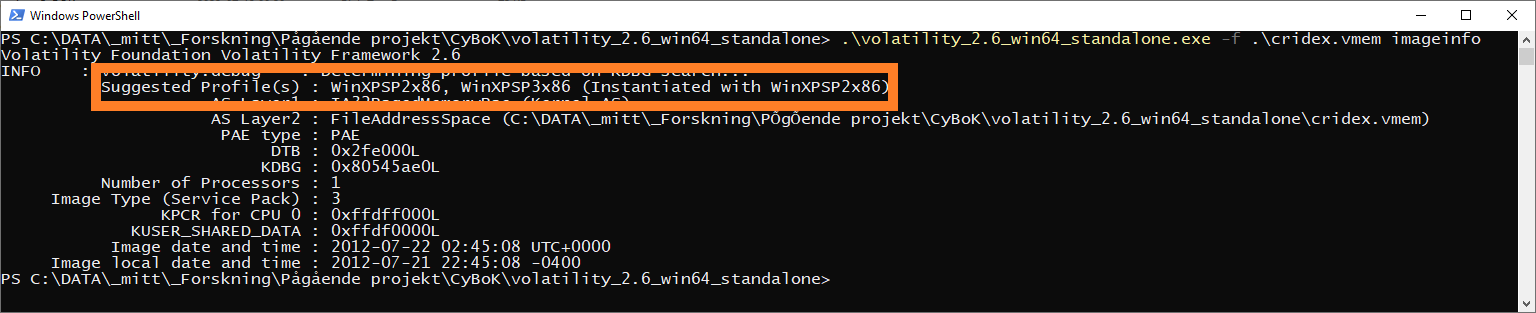
Volatility 2.6 can be best seen as a framework with a set of modules. Each module performs a specific task. Volatility 2.6 is executed by calling the executable using a PowerShell terminal. The basic syntax is (Assuming that vol.py and the memory dump to analyze are both in the current working directory):

.\volatility\_2.6\_win64\_standalone.exe -f *memdumpfilename* -–profile=”*profilename*” *modulename*

The profile name should match the operating system and version that the memory dump is from. This tells Volatility how to read the memory dump and is necessary for Volatility to be able to correctly interpret the memory dump. Volatility can analyze the memory dump and suggest possible profiles to use by issuing the module *imageinfo* as follows.

.\volatility\_2.6\_win64\_standalone.exe -f *memdumpfilename* imageinfo

The output will provide an overview of the memory dump as shown below, the highlighted area shows the suggested profiles.



To get an idea of what Volatility can do, we can resort to the official documentation[[2]](#footnote-2) or use the built-in help-function as follows.

.\volatility\_2.6\_win64\_standalone.exe -h

Now that you know how the tool works, it is all about using different modules to find as much information as possible. The current scenario is Malware identification, and that makes us look for evil processes, bad network connections, and such. However, Memory analysis can also be used to identify misuse and find important information in criminal investigation, and such a case could include looking for encrypted chat messages, encryption keys, or something else. Let’s give it a try!

### Volatility 3 basic usage

Volatility 3 is essentially a Python script that in turn can call other Python scripts. We can call those other script modules. What we essentially need to do is to use python3 to execute Volatility3 and have it use some module. We also need to point to the file containing the memory dump we are to analyze. As such, the basic syntax is (Assuming that vol.py and the memory dump to analyze are both in the current working directory):

Python vol.py -f *memdumpfilename modulename*

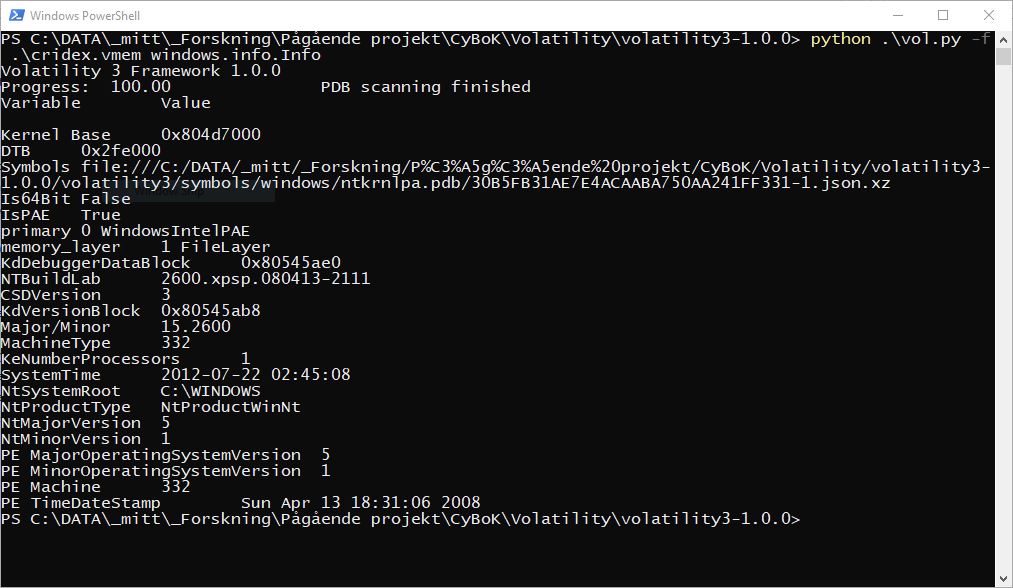
To make Volatility do something, we need to use the different available modules. The modules are listed in the official volatility documentation[[3]](#footnote-3). They can also be listed by invoking the included help function as follows:

Python vol.py -h

A typical starting point in any analysis is to view some default information about the computer the memory dump was taken from. That can be achieved by using the module called info, as follows (Note that modules are case sensitive):

Python vol.py -f .\cridex.vmem windows.info.Info

The output will reveal when the memory dump was created and the system time at that point. It will also reveal that the memory dump is from an NT-based Windows system. It should look as follows:



Now that you know how the tool works, it is all about using different modules to find as much information as possible. The current scenario is Malware identification, and that makes us look for evil processes, bad network connections, and such. However, Memory analysis can also be used to identify misuse and find important information in criminal investigation, and such a case could include looking for encrypted chat messages, encryption keys, or something else. Let’s give it a try!

## Do it yourself

This will be an exercise in finding different information nuggets with the goal of “proving” that the memory dump is indeed from a computer infected by the Cridex malware. In a real-world case, you would not have that information. But since we are currently training, we are entitled to a bit of support. Feel free to use the internet to research the Cridex Malware if you need more information. That could help you understand what to look for. You can choose what Volatility version to use for this task, and you can use them both if that suits your needs. Do the tasks below and try to answer the corresponding questions.

I am using Volatility Version \_\_\_\_\_\_\_

Let’s first try to find out what version of Windows is used by the computer the memory dump is from.

Module used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Listing running processes is a good starting point, it lets us know what processes the computer was running at the time of the memory dump. Find out the number of running processes at the time of the memory dump.

Module used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

On a similar note, what is the Process ID of the lsass.exe process?

Module used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

User accounts can be important, what is the name of the user with RID 1003? *TIP: While this can be solved in different ways, the easiest is to figure out how to read RID from windows registry.*

Module used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It is not uncommon for bad behavior to be executed using the command line. Find out what process has the most command line arguments.

Module used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A malware may want to send away information and needs a network socket. Figure out how many open connections there are.

Module used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

There is also a wonderful function that attempts to identify low-level instructions that could potentially be strange. While the output is verbose, it is well worth a try. Use the malfind module to figure out the processes with potentially harmful instructions.

Module used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Finally, try to work on your own to find good indications of infections in this memory dump. The task is to prove that it is indeed infected with Cridex. Feel free to export data from the memory dump and have your antivirus or some online service analyze it. Also, feel free to use the internet to research more about Cridex.

Modules used:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. https://www.cybok.org/knowledgebase1\_1/ [↑](#footnote-ref-1)
2. https://github.com/volatilityfoundation/volatility/wiki/Command-Reference [↑](#footnote-ref-2)
3. https://volatility3.readthedocs.io/en/stable/volatility3.plugins.html [↑](#footnote-ref-3)