Texas Tech University

A Course on Digital forensics

Memory Forensics

Module 3 – The Volatility Framework

Akbar S. Namin, Spring 2018

CHAPTER 3 – THE VOLATILITY FRAMEWORK

OUTLINE

CONTENT

The Volatility Framework is a completely open collection of tools, implemented in Python under the GNU General Public License 2. Analysts use Volatility for the extraction of digital artifacts from volatile memory (RAM) samples.

1. Why Volatility?

Volatility is not the only memory forensics application—it was specifically designed to be different. Here are some of the reasons why it quickly became our tool of choice:

- ✓ **A single, cohesive framework:** Volatility analyzes memory from 32- and 64-bit systems. Volatility allows it to easily support new operating systems and architectures as they are released.
- ✓ **It is Open Source GPLv2**
- ✓ **It is written in Python**
- ✓ **Runs on Windows, Linux, or Mac analysis systems**

- ✓ **Extensible and scriptable application programming interface (API)**: For example, you can use Volatility to drive your malware sandbox.
- ✓ **Unparalleled feature sets:** Volatility provides functionality that even Microsoft's own kernel debugger doesn't support.
- ✓ **Comprehensive coverage of file formats:** Volatility can analyze raw dumps, crash dumps, hibernation files, and various other formats
- ✓ **Fast and efficient algorithms**
- ✓ **Serious and powerful community**
- ✓ **Focused on forensics, incident response, and malware**

2. What Volatility Is Not

- ✓ **It is not a memory acquisition tool**: Volatility does not acquire memory from target systems.
- ✓ **It is not a GUI**
- ✓ **It is not bug-free**

3. Installation

- With every major release, Volatility is distributed in several formats, including a standalone Windows executable, a Windows Python module installer, and source code packages in both zip and gzip/tarball archives.

4. The Framework

- The Volatility Framework consists of several subsystems that work together to provide a robust set of features.

a. VTypes

- This is Volatility's structure definition and parsing language.
- Because Volatility is written in Python, you need a way to represent C data structures in Python source files. *VTypes* enable you to do exactly that.
- For the upcoming example, assume that you are dealing with C data structure like this:

```
struct process {
int pid;
int parent_pid;
char name[10];
char * command_line;
void * ptv;
};
```

- The equivalent structure in the VType language is as follows:

```
'process' : [ 26, {
'pid' : [ 0, ['int']],
'parent_pid' : [ 4, ['int']],
'name' : [ 8, ['array', 10, ['char']]],
'command_line' : [ 18, ['pointer', ['char']]],
'ptv' : [ 22, ['pointer', ['void']]],
}]
```

b. Generating VTypes

- Despite the fact that working with VType syntax can be a fairly simple task once you practice, the sheer number of structures that an operating system or application uses makes it highly impractical to generate them all by hand.
- The generation of VTypes falls back into the manual realm: Developers or researchers must reverse engineer components of the operating system and create their own structure definitions that can be used with Volatility.

c. Overlays

- *Overlays* enable you to fix up, or patch, the automatically generated structure definitions.
- This is an important aspect of memory forensics because operating system code often makes heavy use of void pointers (void *) in their structures.
- Aside from providing the ability to make structure definitions more accurate, overlays also come in handy for convenience and consistency purposes.

d. Objects and Classes

- A *Volatility object* (or just *object* for short) is an instance of a structure that exists at a specific address within an address space (AS).
- An *object class* enables you to extend the functionality of an object. In other words, you can attach methods or properties to an object that then become accessible to all instances of the object.

e. Profiles

- A profile is a collection of the VTypes, overlays, and object classes for a specific operating system version and hardware architecture (x86, x64, ARM).
- In addition to these components, a profile also includes the following:
  - ✓ Metadata
  - ✓ System call information
  - ✓ Constant values
  - ✓ Native types
  - ✓ System map

f. Address Spaces

- An address space (AS) is an interface that provides flexible and consistent access to data in RAM, handles virtual-to-physical-address translation when necessary, and transparently accounts for differences in memory dump file formats.

g. Virtual/Paged Address Spaces

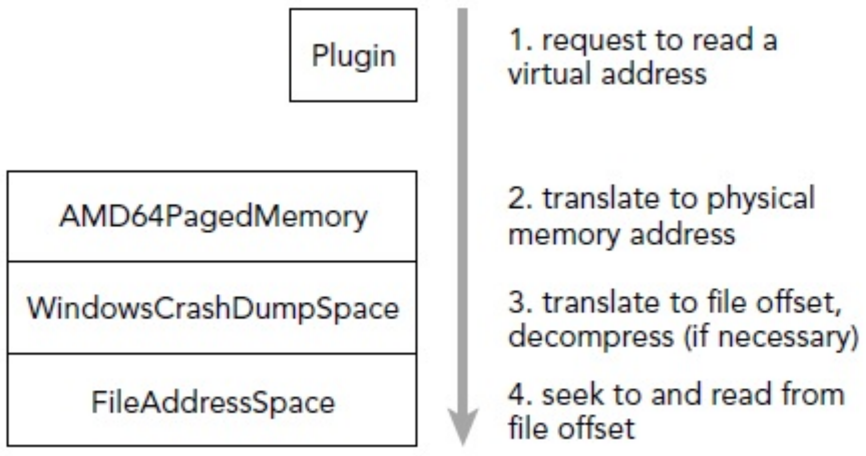
- These ASs provide support for reconstructing virtual memory.
- They use many of the same algorithms that Intel and AMD processors use for address translation, so it's possible to find data in an offline manner, independent from the target operating system from which the memory dump was acquired.

h. Physical Address Spaces

- These ASs mainly deal with the various file formats that memory acquisition tools use to store physical memory.
- This category also includes crash dumps, hibernation files, and VM snapshots (such as VMware saved state and VirtualBox core dumps) that may in fact store vendor-specific metadata along with the actual memory from the target machine.

i. Address Space Stacking

- The few different categories of ASs that you have learned about so far are most often used together in support of each other.


Plugin
AMD64PagedMemory
WindowsCrashDumpSpace
FileAddressSpace
1. request to read a virtual address
2. translate to physical memory address
3. translate to file offset, decompress (if necessary)
4. seek to and read from file offset

Volatility uses a stacked AS

- In this example, you are dealing with a crash dump from a 64-bit Windows system.

j. Plugin System

- Plugins enable you to expand upon the existing Volatility Framework. For example, an *address space plugin* can introduce support for operating systems that run on new CPU chipsets.
- *Analysis plugins* are written to find and analyze specific components of the operating system, user applications, or even malicious code samples.
- Regardless of which type of plugin you write, there are APIs and templates provided by the framework that make it relatively easy to design and integrate your ideas.

k. Core Plugins

- The core Volatility Framework includes over 200 analysis plugins.

5. Using Volatility

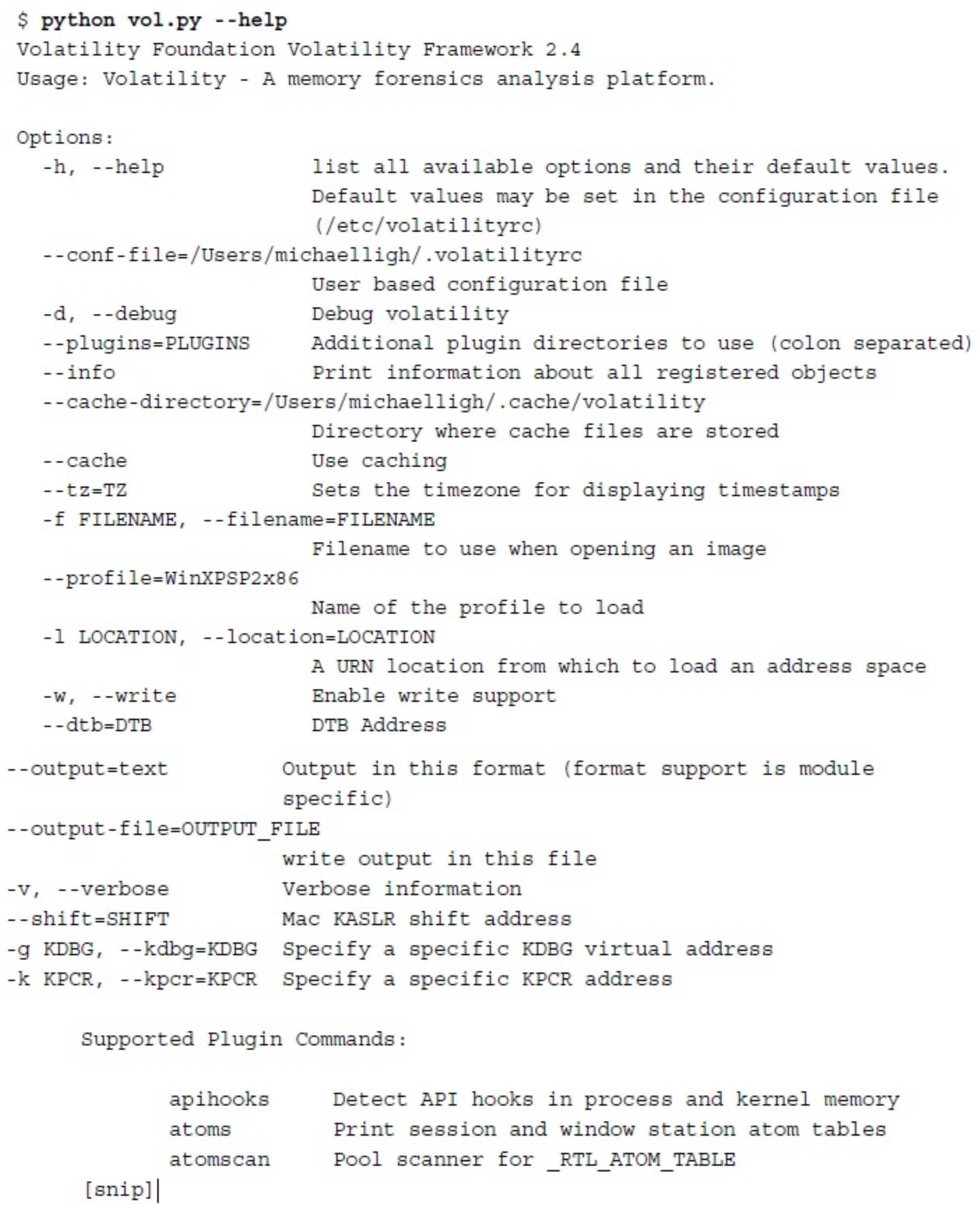
a. Basic Commands

- You execute the main Python script (vol.py) and then pass it the path to your memory dump file, the name of the profile, and the plugin to execute (and optionally plugin-specific parameters):

```
$ python vol.py –f <FILENAME> --profile=<PROFILE> <PLUGIN> [ARGS]
```

b. Displaying Help

- To display the main help menu, pass –h/--help on the command line.



```
$ python vol.py --help
Volatility Foundation Volatility Framework 2.4
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/Users/michaelligh/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --plugins=PLUGINS     Additional plugin directories to use (colon separated)
  --info                Print information about all registered objects
  --cache-directory=/Users/michaelligh/.cache/volatility
                        Directory where cache files are stored
  --cache               Use caching
  --tz=TZ               Sets the timezone for displaying timestamps
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  --profile=WinXPSP2x86
                        Name of the profile to load
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
  -w, --write           Enable write support
  --dtb=DTB             DTB Address
--output=text         Output in this format (format support is module
                      specific)
--output-file=OUTPUT_FILE
                      write output in this file
-v, --verbose         Verbose information
--shift=SHIFT         Mac KASLR shift address
-g KDBG, --kdbg=KDBG  Specify a specific KDBG virtual address
-k KPCR, --kpcr=KPCR  Specify a specific KPCR address

      Supported Plugin Commands:

              apihooks      Detect API hooks in process and kernel memory
              atoms         Print session and window station atom tables
              atomscan      Pool scanner for _RTL_ATOM_TABLE
      [snip]
```

c. Selecting a Profile

- One of the options from the global help menu that you will use most frequently is --profile.

d. Issues with Profile Selection

- Volatility provides a --profile option in the first place.

e. Alternatives to command-Line Options

f. Controlling Plugin Output