

Memory Analysis with DumpIt and Volatility

By Russ McRee – ISSA member, Puget Sound (Seattle), USA Chapter

Join the Discussion
Connect



Prerequisites

SIFT 2.1 if you'd like a forensics-focused virtual machine with Volatility ready to go

Python version 2.6 or higher on Window, Linux, or Mac OS X

Some plugins require third party libraries¹



Sept. 11, 2001: "To honor those whose lives were lost, their families, and all who sacrifice that we may live in freedom. We will never forget."

Two recent releases give cause for celebration and discussion in toolsmith. First, in July, Matthieu Suiche of MoonSols released DumpIt2 for general consumption, a "fusion of win32dd and win64dd in one executable." Running DumpIt on the target system generates a copy of the physical memory in the current directory. That good news was followed by Ken Pryor's post on the SANS Computer Forensics Blog³ (I'm a regular reader, you should be too) mentioning the fact that Volatility 2.0 had been released in time for the Open Memory Forensics Workshop, and that SIFT 2.14 was also available. Coincidence? I think not. Volatility 2.0 is available on SIFT 2.1. Thus, the perfect storm formed, creating the ideal opportunity to discuss the complete life cycle of memory acquisition and analysis for forensics and incident response. In May 2010, we discussed SIFT 2.0 and mentioned how useful Volatility is, but didn't give its due. Always time to make up for our shortcomings, right?

If you are already aware of Volatility, "the Volatility Framework is a completely open collection of tools, implemented in Python under the GPL, for the extraction of digital artifacts from volatile memory (RAM) samples."⁵

One thing I've always loved about writing *toolsmith* is meeting people (virtually or in person) who share the same passion for and dedication to our discipline. Such is the case with the Volatility community.

As always, I reached out to project leads/contributors and benefited from very personal feedback regarding Volatility.

Mike Auty and Michael Hale Ligh (MHL) each offered valuable insight you may not glean from the impressive technical documentation available to Volatility users.⁶

Regarding the Volatility roadmap, Mike Auty indicated that the team has an ambitious goal for their next release (which they want to release in six months, a big change from their last release). They're hoping to add Linux support (as written by Andrew Case), as well as 64-bit support for Windows (still being written), and a general tidy up for the code base without breaking the API.

MHL offered the following:

"At the Open Memory Forensics Workshop (OMFW)⁷ in late July, many of the developers sat on a panel and described what got them involved in the project. Some of us are experts in disk forensics, wanting to extend those skills to memory analysis. Some are experts in forensics for platforms other than Windows (such as Linux, Android, etc.) who were looking for a common platform to integrate code. I personally was looking for new tools that could help me understand the Windows kernel better and make my training course on rootkits more interesting to people already familiar with running live tools such as GMER, IceSword, Rootkit Unhooker, etc., I think the open source nature of the project is inviting to new-comers, and I often refer to the source code as a Python version of the Windows Internals book, since you can really learn a lot about Windows by just looking at how Volatility enumerates evidence."

Man, does that say it all! Stay with this thinking and consider this additional nugget of Volatility majesty from MHL. In his blog post specific to using Volatility to detect Stuxnet, *Stuxnet's Footprint in Memory with Volatility 2.0*,⁸ he discusses Sysinternals tools side-by-side with artifacts identified with Volatility. MHL is dead on right when he says this may "interest your readers, especially those who have never heard of Volatility before, because it builds on something they do know - Sysinternals tools."

This was an incredibly timely post for me as I read it right on the heels of hosting the venerable Mark Russinovich at the ISSA Puget Sound July chapter meeting where he presented Zero Day Malware Cleaning with the Sysinternals Tools, including live analysis of the infamous Stuxnet virus.

1 http://code.google.com/p/volatility/wiki/FAQ#What_are_the_dependencies_for_running_Volatility?

2 <http://www.moonsols.com/ressources>.

3 <http://computer-forensics.sans.org/blog>.

4 <http://computer-forensics.sans.org/community/downloads>.

5 <https://www.volatilitysystems.com/default/volatility#overview>.

6 <https://code.google.com/p/volatility/w/list>.

7 <https://www.volatilitysystems.com/default/omfw>.

8 <http://mnin.blogspot.com/2011/06/examining-stuxnets-footprint-in-memory.html>.

Figure 1 – Run DumpIt

```

C:\tools\DumpIt\DumpIt.exe
DumpIt - v1.3.2.20110401 - One click memory memory dumper
Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>

Address space size:      1073741824 bytes <  1024 Mb>
Free space size:        1996926976 bytes <  1904 Mb>

* Destination = \\??\C:\tools\DumpIt\HIOMALVM02-20110822-041239.raw
--> Are you sure you want to continue? [y/n] _

```

See how this all comes together so nicely?

Read Mark's three posts on Technet⁹ followed immediately by MHL's post on his MNIN Security Blog, then explore Volatility for yourself. I'll offer you some SpyEye analysis examples below.

NOTE: MHL was one of the authors of *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*; I'll let the reviews speak for themselves (there are ten reviews on Amazon and all are five stars). I share Harlan's¹⁰ take on the book and simply recommend that you buy it if this topic interests you.

Some final thoughts from Aaron Walters, the principal developer and lead for Volatility:

"We have a hard-working development team and it's appreciated when people recognize the work that is being done. The goal was to build a modular and extendable framework that would allow researchers and practitioners to come together and collaborate, as a result, shortening the amount of time it takes to get cutting-edge research into the hands of practitioners. We also wanted to encourage and push the technical advancement of the digital forensics field which had frequently lagged behind the offensive community. It's amazing to see how far the project has come since I dropped the initial public release more than four years ago. With the great community now supporting the project, there are lot more exciting enhancements in the pipe line..."

DumpIt

Before you can conduct victim system analysis you need to capture memory. Some form of dd, including MoonSols win32dd and win64dd were/are de facto standards but the recently released MoonSols DumpIt makes the process incredibly simple.

On a victim system (local or via psexec) running DumpIt is as easy as executing DumpIt.exe from the command-line or Windows Explorer. The raw memory dump will be generated and written to the same directory you're running DumpIt from; answer *yes* or *no* when asked if you wish to continue, and that's all there is to it. A .raw memory image named for the hostname, date, and UTC time will result (Figure 1). DumpIt is ideal for your incident response jump kit; deploy

the executable on a USB key or your preferred response media.

Painless and simple, yes? I ran DumpIt on a Windows XP SP3 virtual machine that had been freshly compromised with SpyEye (md5: 00B77D6087F00620508303ACD3FD846A), an exercise that resulted in my being swiftly shunted by my DSL provider.

Their consumer protection program was kind enough to let me know that "malicious traffic was originating from my account." Duh, thanks for that, I didn't know. :-)

Clearly, it's time to VPN that traffic out through a cloud node, but I digress.

SpyEye has been in the news again lately with *USA Today Tech* describing a probable surge in SpyEye attacks due to increased availability and reduced cost from what used to be as much as \$10,000 for all the bells and whistles, down to as little as \$95 for the latest version.¹¹ Sounds like a good time for a little SpyEye analysis, yes?

I copied the DumpIt-spawned .raw image from the pwned VM to my shiny new SIFT 2.1 VM and got to work.

Volatility 2.0

So much excellent documentation exists for Volatility; on the wiki I suggest you immediately read the FAQ, Basic Usage, Command Reference, and Features By Plugin.

As discussed in May 2010's *toolsmith* on SIFT 2.0, you can make use of Volatility via PTK, but given that we've discussed that methodology already and the fact that there are constraints imposed by the UI, we're going to drive Volatility from the command line for this effort. My memory image was named HIOMALVM02-20110811-165458.raw by DumpIt; I shortened it to HIOMALVM02.raw for ease of documentation and word space.

I executed `vol.py imageinfo -f HIOMALVM02.raw` to confirm just that, image information. This plugin provided PAE (physical address extension) status as well as hex offsets for DTB (Directory Table Base), KDBG (short for `_KDDEBUGGER_DATA64`), KPCR (Kernel Processor Control Region), time stamps and processor counts (Figure 2).

Windows XP SP3, check.

Runtime analysis of my SpyEye sample gave me a few queryable entities to throw at Volatility for good measure, but we'll operate here as if the only information we have is only suspicion of system compromise.

It's always good to see what network connections may have been made.

```
vol.py --profile=WinXPSP3x86 connscan -f HIOMALVM02.raw
```

9 <http://blogs.technet.com/b/markrussinovich/archive/2011/03/30/3416253.aspx>.

10 <http://windowsir.blogspot.com/2010/12/book-review-malware-analysts-cookbook.html>.

11 <http://www.usatoday.com/tech/news/story/2011/08/SpyEye-hacker-toolkit-to-lead-to-surge-in-cyberattacks/50080368/1>.

Figure 2 – Imageinfo plugin results

```
sansforensics@SIFT-Workstation:~/Desktop/cases/HIOMALVM02$ vol.py imageinfo -f HIOMALVM02.raw
Volatile Systems Volatility Framework 2.1_alpha
Suggested Profile(s) : WinXPSP3x86, WinXPSP2x86 (Instantiated with WinXPSP2x86)
AS Layer1 : JKIA32PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/cases/HIOMALVM02/HIOMALVM02.raw)
PAE type : No PAE
DTB : 0x39000
KDBG : 0x8054ce60L
KPCR : 0xfffff000L
KUSER_SHARED_DATA : 0xfffff000L
Image date and time : 2011-08-11 16:55:05
Image local date and time : 2011-08-11 16:55:05
Number of Processors : 1
Image Type : Service Pack 3
```

The *connscan* plugin scans physical memory for connection objects.

Results included:

Offset	Local Address	Remote Address	Pid
0x0629e5d8	192.168.248.19:1705	82.165.218.111:80	3376
0x063c9008	192.168.248.19:1381	188.40.138.148:80	1512

Interesting, both IPs are in Germany. My VMs don't make known good connections to Germany, so let's build from here.

The PID associated with the second connection to 188.40.138.148 over port 80 is 1512.

The *pslist* plugin prints active processes by walking the PsActiveProcessHead linked list.

```
vol.py --profile=WinXPSP3x86 pslist -P -f HIOMALVM02.raw
```

Use -P to acquire the physical offset for a process, rather than virtual, which is default.

Results included a number of PPID (parent process IDs) that matched the 1512 PID from *connscan*:

Offset(P)	Name	PID	PPID	Thds	Hnds	Time
0x8645d6e8	WinPatrol.exe	1864	1512	4	121	2011-08-06 19:51:19
0x865aa220	VMwareUser.exe	1908	1512	7	230	2011-08-06 19:51:20
0x8634f4c0	msseces.exe	1916	1512	10	248	2011-08-06 19:51:20
0x865b11d8	ctfmon.exe	1924	1512	2	100	2011-08-06 19:51:20
0x863be900	PrintScreen.exe	1932	1512	3	59	2011-08-06 19:51:21
0x861a6020	cleansweep.exe	3328	1512	0	---	2011-08-11 05:25:18
0x86175ca0	DumpIt.exe	3216	1512	3	45	2011-08-11 16:54:58

I highlighted the process that jumped out at me given the anomalous time stamp, a 0 thread count and no handles.

Let's check for additional references to *cleansweep*.

The *ptree* plugin prints the process list as a tree so you can visualize the parent/child relationships.

```
vol.py --profile=WinXPSP3x86 ptree -f HIOMALVM02.raw
```

Results included the PPID of 1512, and the Pid for *cleansweep*.

Name	Pid	PPid	Thds	Hnds	Time
0x8657C9F8:					
explorer.exe	1512	1472	16	634	2011-08-06 19:51:17
.0x861A6020:					
cleansweep.exe	3328	1512	0	---	2011-08-11 05:25:18

Ah, the victim most likely downloaded *cleansweep.exe* and executed it via Windows Explorer.

But can we extract actual binaries for analysis via the like of Virus Total? Of course.

This is where the malware plugins are very helpful. I already know I'm not going to have much luck exploring PID 3328 as it has no threads or open handles. MHL points out¹² that a process such as *cleansweep.exe* typically can't remain active with 0 threads as a process is simply a container for threads, and it will terminate when the final thread ex-

its. *Cleansweep.exe* is still in the process list probably because another component of the malware (likely the one that started *cleansweep.exe* in the first place) never called *CloseHandle* to properly "clean up." That said, the PPID of 1512 has clearly spawned PID 3328, so let's explore the PPID with the *malfind* plugin, which extracts injected DLLs, injected code, unpacker stubs, and API hook trampolines. The malware (*malfind*) plugins don't come packaged with *volatility*, but are in fact a part of the above mentioned *Malware Analyst's Cookbook*; the latest version can also be downloaded.¹³

```
vol.py --profile=WinXPSP3x86 -f HIOMALVM02.raw malfind -p 1512 -D output/ yielded PE32 gold as seen in Figure 3 (next page).
```

Malfind dropped each of the suspicious PE files it discovered to my output directory as *.dmp* files. I submitted each to Virus Total, and bingo, all three were malicious and identified as *SpyEye* variants as seen in Figure 4 (next page).

In essence, we've done for ourselves via memory analysis what online services such as Threat Expert will do via runtime analysis. Compare this discussion to the Threat Expert results¹⁴ for the *SpyEye* sample I used.

There is so much more I could have discussed here, but space is limited and we've pinned the VU meter in the red, so go read the *Malware Cookbook* as well as all the online *Volatility* resources, and push *Volatility* to the boundaries of your skillset and imagination. In my case the

only limiting factors were constraints on my time and my lack of knowledge. There are few limits imposed on you by *Volatility*; 64bit and Linux analysis support are pending. Get to it!

In conclusion

I've said it before and I'll say it again. I love *Volatility*. *Volatility 2.0* makes me squeal with delight and clap my hands like a little kid at the state fair. Oh the indignity of it all, a grown man cackling and clapping when he finds the resident evil via a quick memory image and the glorious volatile memory analysis framework that is *Volatility*.

12 <http://mmin.blogspot.com/2011/03/mis-leading-active-in.html>.

13 <http://code.google.com/p/malwarecookbook/source/browse/trunk/malware.py>.

14 <http://www.threatexpert.com/report.aspx?md5=00b77d6087f00620508303acd3fd846a>.

Figure 3 – Malfind plugin results

```
sansforensics@SIFT-Workstation:~/Desktop/cases/HIOMALVM02$ vol.py --profile=WinXP5
-D output/
Volatile Systems Volatility Framework 2.1_alpha
Name      Pid      Start      End      Tag      Hits      Protect
explorer.exe 1512    0xea00000 0xea26fff0 VadS      0          PAGE_EXECUTE_REA
Dumped to: output/explorer.exe.657c9f8.0ea00000-0ea26fff.dmp
0xea00000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00
0xea00010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
0xea00020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xea00030 00 00 00 00 00 00 00 00 00 00 00 00 e0 00 00 00
0xea00040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68
0xea00050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f
0xea00060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20
0xea00070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00

explorer.exe 1512    0xea50000 0xea7afff0 VadS      0          PAGE_EXECUTE_REA
Dumped to: output/explorer.exe.657c9f8.0ea50000-0ea7afff.dmp
0xea50000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00
0xea50010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
0xea50020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xea50030 00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00
0xea50040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68
0xea50050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f
0xea50060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20
0xea50070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00

explorer.exe 1512    0xeaa0000 0xeaccfff0 VadS      0          PAGE_EXECUTE_REA
Dumped to: output/explorer.exe.657c9f8.0eaa0000-0eaccfff.dmp
0xeaa0000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00
0xeaa0010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
0xeaa0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xeaa0030 00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00
0xeaa0040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68
0xeaa0050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f
0xeaa0060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20
0xeaa0070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00
```

Do you really need any more motivation to explore and use Volatility for yourself?

There's a great list of samples¹⁵ to grab and play with. Do so and enjoy! As it has for me, this process will likely become inherent to your IR and forensic efforts, perhaps even surpassing other tactics and methods as your preferred, go-to approach.

Ping me via email if you have questions (russ at holisticinfosec dot org).

Cheers...until next month.

Acknowledgements

—Mike Auty & Michael Hale Ligh of the Volatility project.

—Aron Walters – Volatility lead

About the Author

Russ McRee, GCIH, GCEA, GPEN, CISSP, is team leader and senior security analyst for Microsoft's Online Services Security Incident Management team. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at [russ at holisticinfosec dot org](mailto:russ@holisticinfosec.org).

An earlier comment from MHL bears repeating here. Volatility source code can be likened to “a Python version of the Windows Internals book, since you can really learn a lot about Windows by just looking at how Volatility enumerates evidence.” Yeah, what he said.

15 http://code.google.com/p/volatility/wiki/FAQ#Are_there_any_public_memory_samples_available_that_I_can_use_for.

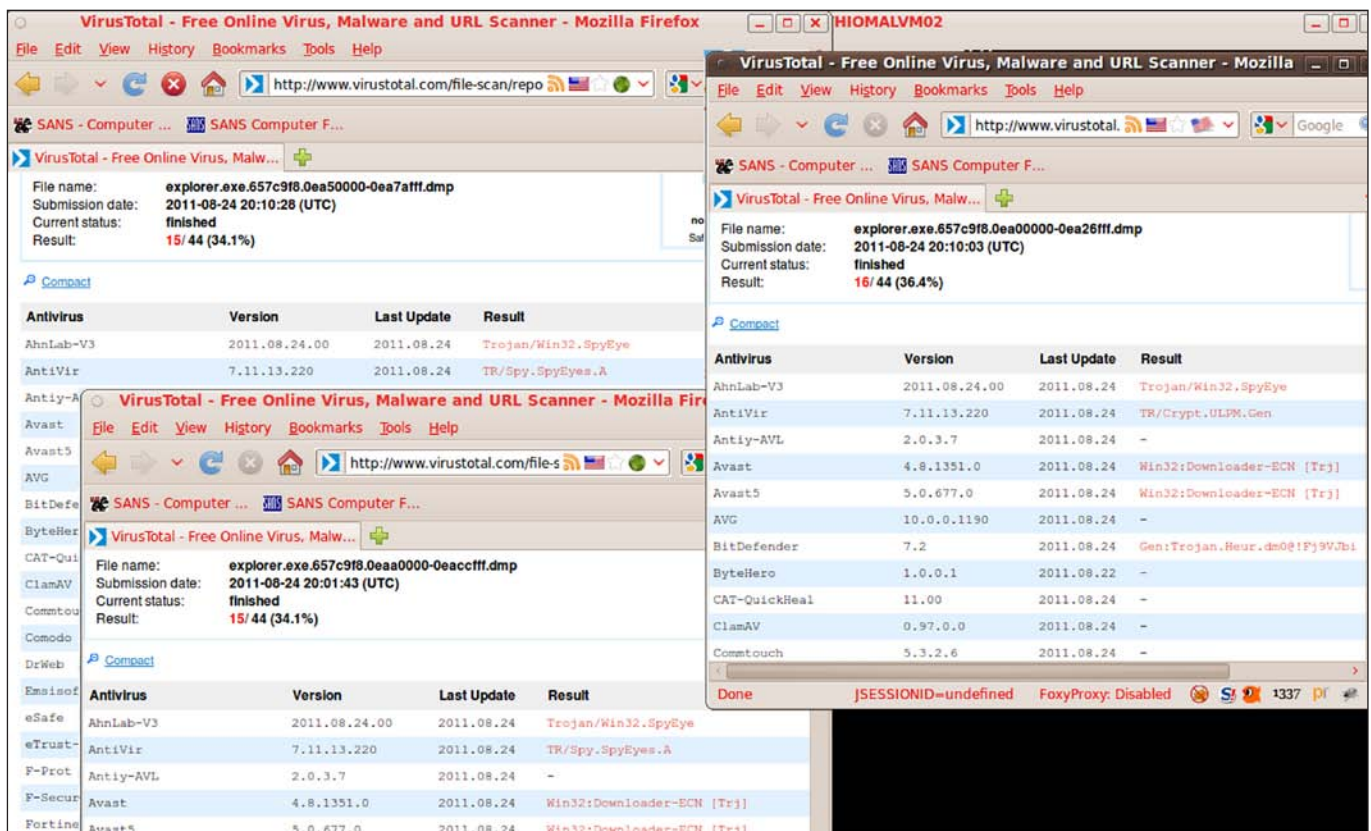


Figure 4 – PE results from Virus Total