**PowerShell Volatility Scripts.**

Tom Willett

1/15/2015

Here are some instructions and hints to help you use my Volatility scripts for memory analysis.

Some general PowerShell instruction.

1. You will need to enable scripting in PowerShell to use these scripts. I prefer to set it to RemoteSigned rather than Unrestricted – it still gives you a little protection. PowerShell is a big security hole – don’t get me started. You enable scripting by issuing the command

Ps: set-executionpolicy remotesigned

You must be in a PowerShell window run as administrator or it will complain about this command. Hint. PowerShell has great command completion so with the previous command I would just type set-ex and hit tab. It will also do file globbing the same way.

1. You may still not be able to run scripts because Windows will detect that they came from the Internet. If so just unblock them. From file explorer right click on the script and choose unblock, or from PowerShell

Ps: unblock-file .\vol.ps1

1. All PowerShell scripts must be given an explicit path to execute. Thus even if you are in the same directory as the script you must give it an explicit path, e.g. .\vol.ps1 or z:\powershell\vol.ps1.
2. I always suggest putting your PowerShell files in one place and setting PowerShell to always open in this place. To do that open up your profile (type $profile at the PowerShell prompt to see the location – you will probably have to create the directories and file) and put in the command set-location c:\powershell or wherever you put your files. The profile file is like your .cshrc in linux, you can load libraries like the AD extensions every time you open PowerShell, configure most anything and run any PowerShell command.
3. Most of the common linux command are aliased in PowerShell, e.g. I naturally type ls instead of dir and it just works. Help is also available from the command line. The specific command is get-help but it is aliased to man and that is what I usually type. All my scripts have built in help so you can type:

Ps: man .\vol.ps1

And it will display the built in help. Add –examples to the man command and it will show you command examples.

**VOL.PS1**

Vol.ps1 runs a series of Volatility commands on a memory image. From the built in help, here are some execution examples:

Ps: .\vol.ps1 c:\images\image.mem

Runs the script against c:\images\image.mem, checks what kind of image it is and allows you to choose. It puts the output in c:\images\VolatilityOutput-image.mem

Ps: .\vol.ps1 c:\images\image.mem WinXPSP3x86

Runs the script against c:\images\image.mem and uses WinXPSP3x86 as the image type. It puts the output in c:\images\VolatilityOutput-image.mem

In the first example, it will run the volatility imageinfo command first and present its best guesses for the system type and let you choose the correct one (if in doubt, choose the last one). In the second example, the specific system type was submitted on the command line.

Vol.ps1 assumes that the standalone version of volatility 2.4 available at <http://www.volatilityfoundation.org/#!24/c12wa> , a helper script .\split.ps1 and the Sysinternals strings utility <http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx> are in the same directory as the script.

It runs the following regular commands against the image: hivelist, userassist, pslist, psscan, pstree, psxview, modscan, mftparser, ldrmodules, driverscan, driverirp, devicetree, unloadedmodules, envars, dlllist, getsids, handles, filescan, svcscan, cmdscan, and consoles. On a XP class machine it will also run connections, connscan, sockscan, and sockets. On a Win7 class machine it will run netscan and getservicesids. Then it will run the following commands which extract images from the memory image: malfind, dlldump, moddump and procdump. All of these are configurable in variables at the start of the script. You can change the order in which they are run and add and remove commands there. After the volatility modules, the strings utility is ran against the memory image and then it is split into chunks that can be edited if you want to manually view them. Strings is also run against all of the extracted images.

All of the output is put into a directory named \VolatilityOutput- followed by the image name. All the extracted images are put in their own directories under the main directory.

Notes:

You do not have to wait for the script to finish before beginning the analysis. As each module is ran, the output will be put in the Volatility output directory.

Sometimes volatility seems to hang. When this happens you can hit ^c and restart the script. It keeps track of where it is and will skip the modules it has already done and resume where you interrupted it.

The script keeps track of the system type. Once you enter it you will not have to enter it again.

The script also turns on buffering to speed things up. The buffer is placed under the volatility output directory, moved from its default location in you profile, so that all the evidence is kept together.

**Voltimeline.ps1**

Like the main vol.ps1 script, the voltimeline.ps1 script runs multiple timeline commands (timeliner, shellbags and mftparser) against a memory image to gather a more complete timeline. The mftparser will often return mft entries not written to disk or deleted. A helper script mactime.ps1 is required. The Sleuthkit mactime script seemed to choke on the massive timeline created.

A note about times. The times extracted from the memory image are UTC. You can convert these to whatever timezone you wish with the TZ parameter. The timezones are the standard Windows Timezones. You can list them by issuing the following command

Ps: [system.timezoneinfo]::getsystemtimezones() | out-gridview

Use the value from the ID column.

Some examples of voltimeline.ps1 usage:

PS D:\> .\VolTimeline.ps1 d:\image.mem

Produce a memory timeline for d:\image.mem. If the vol.ps1 script has been run, the image type from that run is used. Otherwise, the imageinfo plugin is run against the image. The image type is saved for future use.

PS D:\> .\VolTimeline.ps1 d:\image.mem WinXPsp2

Produce a memory timeline for d:\image.mem using the system type of WinXPsp2. The image type is saved for future use.

PS D:\> .\VolTimeline.ps1 d:\image.mem –tz “Eastern Standard Time”

Produce a memory timeline for d:\image.mem and convert the times to Eastern Standard Time. If the vol.ps1 script has been run, the image type from that run is used. Otherwise, the imageinfo plugin is run against the image. The image type is saved for future use.

The output of the script is a csv file named (Imagename)-timeline.csv. So using the above examples the output would be image.mem-timeline.csv placed in the same directory as the image.

This script takes a while to run – I would suggest letting it run overnight.

2/22/2015

**VolLinux.ps1**

Linux memory images are different from windows memory images. There is an entirely different set of commands and you have to create a profile specifically for each version and kernel. See <https://code.google.com/p/volatility/wiki/LinuxMemoryForensics> for information about creating a Linux profile. Because the Linux profiles are different, the script cannot run the imageinfo command beforehand to determine the image type. You must provide the profile and name of the image. A Linux profile is a zip file containing information on the kernel's data structures and debug symbols. The script expects to find this profile in a directory called profiles directly under the directory from where the script is run.

The Linux script runs these commands against the image: linux\_pslist, linux\_psaux, linux\_pstree, linux\_pslist\_cache, linux\_pidhashtable, linux\_psxview, linux\_lsof, linux\_memmap, linux\_proc\_maps, linux\_bash, linux\_lsmod, linux\_check\_afinfo, linux\_check\_tty, linux\_keyboard\_notifier, linux\_check\_creds, linux\_check\_fop, linux\_check\_idt, linux\_check\_syscall, linux\_check\_modules, linux\_check\_creds, Networking, linux\_arp, linux\_ifconfig, linux\_route\_cache, linux\_netstat, linux\_pkt\_queues, linux\_sk\_buff\_cache, linux\_mount, linux\_tmpfs, linux\_moddump and strings. Like vol.ps1, it places the output of these commands in a directory in the same location as the image. You can begin examining the output soon after you start the script. The long running commands are called last.