Vola%lity

OSDFC 2015
Volatility 2.5

• Unified Output
  – Improves how you interface with Volatility (command line, web, thick UI)
  – Extends options for what to do with Volatility output (store, inspect, morph/label)

• Community Integration
  – Instantly sync plugins from various authors around the world

• New OS Profiles
  – Keeping up to date with Windows, Mac, and Linux
Unified Output

- Users: flexibility to specify how you want your output
- Plugin writers: less code results in more functionality
- Framework developers: easy APIs (JSON, DB, etc.)
  - Sqlite3
  - JSON
  - HTML
  - XLSX or CSV
  - Body (pipe delimited)
  - Dot graphs
  - Plain text

Unified Output: Options

$ python vol.py pslist -h
Volatility Foundation Volatility Framework 2.5
Usage: Volatility - A memory forensics analysis platform.

[snip]

Module Output Options: dot, html, json, quick, quicksqlite, sqlite, text, xlsx

----------------------------------------
Module PSList
----------------------------------------
Print all running processes by following the EPROCESS lists
Unified Output: HTML

$ python vol.py -f <FILE> pslist --output=html --output-file=pslist.html

Volatility Foundation Volatility Framework 2.5

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2181173712</td>
<td>userinit.exe</td>
<td>1836</td>
<td>624</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2012-04-28 02:20:55 UTC+0000</td>
<td>2012-04-28 02:22:05 UTC+0000</td>
</tr>
<tr>
<td>2181210144</td>
<td>alg.exe</td>
<td>1880</td>
<td>672</td>
<td>5</td>
<td>102</td>
<td>0</td>
<td>0</td>
<td>2012-04-28 01:56:53 UTC+0000</td>
<td>2012-04-28 01:56:37 UTC+0000</td>
</tr>
<tr>
<td>2182193184</td>
<td>smss.exe</td>
<td>360</td>
<td>4</td>
<td>3</td>
<td>19</td>
<td>-1</td>
<td>0</td>
<td>2012-04-28 01:56:37 UTC+0000</td>
<td>2012-04-28 01:56:37 UTC+0000</td>
</tr>
</tbody>
</table>
Unified Output: JSON

$ python vol.py -f <FILE>--output=json --output-file=pslist.json
Volatility Foundation Volatility Framework 2.5

$ python -m json.tool < pslist.json
{
  "columns": ["Offset(V)", "Name", "PID", "PPID", "Thds", "Hnds", "Sess", "Wow64", "Start", "Exit"],
  "rows": [
    [2182193184, "smss.exe", 360, 4, 3, 19, -1, 0, "2012-04-28 01:56:37 UTC+0000", ""
  ],
}
Unified Output: Sqlite3

$ python vol.py -f <FILE> pslist --output=sqilte --output-file=pslist.db
Volatility Foundation Volatility Framework 2.5

$ sqlite3 pslist.db
SQLite version 3.7.13 2012-07-17 17:46:21
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select * from PSList limit 5;
1|0|2185005104|System|4|0|51|269|-1|0|
2|0|2182193184|smss.exe|360|4|3|19|-1|0|2012-04-28 01:56:37 UTC+0000|
3|0|2182255136|csrss.exe|596|360|11|340|0|0|2012-04-28 01:56:38 UTC+0000|
4|0|2182692896|winlogon.exe|624|360|17|535|0|0|2012-04-28 01:56:39 UTC+0000|
5|0|2182374496|services.exe|672|624|15|238|0|0|2012-04-28 01:56:39 UTC+0000|
Unified Output: Framework Authors

import copy, libapi # see contrib/library_example
import volatility.plugins.taskmods as taskmods
import volatility.plugins.filescan as filescan

config = libapi.get_config("/samples/mem.dmp", "Win7SP1x64")

plugins = (taskmods.PSList, filescan.FileScan)

for plugin in plugins:
    data = libapi.get_json(copy.deepcopy(config),
                           taskmods.PSList)
Community Repo

- Central repository easily links to your Volatility installation
- 40+ plugins from 25+ authors
- https://github.com/volatilityfoundation/community
Community Repo: Setup

1. Git clone the Volatility repository or Download a *Release
2. Git clone the Community repository to $PLUGINSPATH
3. Pass --plugins=$PLUGINSPATH to Volatility when you run it

* Instructions also apply to the standalone exes
Community Repo: FYI

• Plugins only...does not include other frameworks (i.e. Cuckoo, Evolve, GVol)
• Options may conflict
  • Use --plugins=$PLUGINS_PATH/subdir
• Some plugins have multiple homes
  – We will try to keep this one recent/updated
New OS Profiles: Windows

• Windows 10 (Win10x86, Win10x64)
  ✔ Process plugins (handles, sids, privileges, dlls, vads)
  ✔ PE plugins (dlldump, procdump, moddump)
  ✔ Scanning plugins (processes, files, mutexes)
  ✔ Registry plugins (printkey)
  ✔ Misc plugins (services)
  ❏ Isolated user mode memory
  ❏ Compressed memory
  ❏ Hibernation files
New OS Profiles: Mac & Linux

• Mac 10.11 (El Capitan)
• Linux Kernels 4.2.3
Volatility Plugin Contest (2015)

• 12 submissions evaluated based on creativity, usefulness, effort, completeness, and clarity of documentation

• 1\textsuperscript{st} place: $1500$ cash or free 5-day Windows Malware and Memory Forensics Training
• 2\textsuperscript{nd} place: $500$ cash
• 3\textsuperscript{rd} place: $250$ cash
• 4\textsuperscript{th} and 5\textsuperscript{th} place: swag
(1st) Shimcache Memory Scan

- Fred House, Andrew Davis, Claudiu Teodorescu (Mandiant/FireEye)
- Evidence of program execution *before* the system is rebooted
- Supports Windows XP SP2 through Windows 8 and Server 2012 R2
- Applicable to almost any/all types of cases
<table>
<thead>
<tr>
<th>Order</th>
<th>Last Modified</th>
<th>Last Update</th>
<th>Exec Flag</th>
<th>File Size</th>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2015-03-12 16:44:52</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Users\Administrator\Desktop\DumpIt.exe</td>
</tr>
<tr>
<td>2</td>
<td>2013-08-22 11:45:14</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\wbem\WmiPrvSE.exe</td>
</tr>
<tr>
<td>3</td>
<td>2013-08-22 11:44:42</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\wbem\WMIADAP.exe</td>
</tr>
<tr>
<td>4</td>
<td>2013-08-22 11:03:41</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\wundll32.exe</td>
</tr>
<tr>
<td>5</td>
<td>2012-05-01 20:12:56</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Program Files\VMware\VMware Tools\TPAutoConnect.exe</td>
</tr>
<tr>
<td>6</td>
<td>2013-08-22 12:35:25</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\dllhost.exe</td>
</tr>
<tr>
<td>7</td>
<td>2013-08-22 12:39:50</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\OpenWith.exe</td>
</tr>
<tr>
<td>8</td>
<td>2012-11-01 01:05:12</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Program Files\Common Files\VMware\Drivers\vss\comreg.exe</td>
</tr>
<tr>
<td>9</td>
<td>2013-08-22 12:42:49</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\taskhost.exe</td>
</tr>
<tr>
<td>10</td>
<td>2012-05-01 20:12:56</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe</td>
</tr>
<tr>
<td>11</td>
<td>2013-08-22 10:01:15</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\net1.exe</td>
</tr>
<tr>
<td>12</td>
<td>2013-08-22 10:01:50</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\net.exe</td>
</tr>
<tr>
<td>13</td>
<td>2013-08-22 11:01:57</td>
<td></td>
<td>True</td>
<td></td>
<td>SYSVOL\Windows\System32\ThumbnailExtractionHost.exe</td>
</tr>
</tbody>
</table>
(2nd) James Habben: Evolve

• Web interface with AJAX, JQuery, JSON
• Run multiple plugins at once
• Query, filter, and sort the database entries from the web page
• Use or add “morphs” (color coding) to associate countries to IPs, highlight files not in NSRL, etc.
(3rd) Philip Huppert: VM Live Migration

• Thesis: Virtual Machine Introspection During Live Migration
• Full control over the VM with access to migration network traffic
  – No tools installed on ESX or VM
• Offensive/penetration testing
• Forensics
• Scan a VM for malware during the migration
Step 1: Capture ESX Traffic
Step 2: Extract RAM from PCAP

$ ./extract.py winxp_sp3_x86_256mb_vmotion_esxi6.pcap
Processing 192.168.088.010.08000-192.168.088.011.12698
Found VMotion migration in 192.168.088.010.40165-192.168.088.011.08000
Saving to /home/philip/
192.168.088.010.40165-192.168.088.011.08000.vmig
Processing 192.168.088.010.25811-192.168.088.011.08000
Processing report.xml

$ mv 192.168.088.010.40165-192.168.088.011.08000.vmig winxp.vmig
(4th) Ying Li: Python Strings and SSH Keys

• Presented at PyCon 2015
• Extract and contextualize strings within a Python process
• Map strings back to key-value pairs in a dictionary
• Recover RSA keys from the heap of an ssh-agent
• https://www.youtube.com/watch?v=tMKXcc2-xO8
Python Strings and SSH Keys

$ python vol.py -f <FILE> linux_python_strings
Volatility Foundation Volatility Framework 2.5

<table>
<thead>
<tr>
<th>Pid</th>
<th>Name</th>
<th>Size</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>8414</td>
<td>python</td>
<td>48</td>
<td>This is a python string [...]</td>
</tr>
<tr>
<td>8414</td>
<td>python</td>
<td>21</td>
<td>this is another value</td>
</tr>
<tr>
<td>8414</td>
<td>python</td>
<td>15</td>
<td>this is the key</td>
</tr>
</tbody>
</table>

$ python vol.py -f <FILE> linux_ssh_keys --dump-dir /tmp
Volatility Foundation Volatility Framework 2.5

<table>
<thead>
<tr>
<th>Pid</th>
<th>Name</th>
<th>Found-Key Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>8394</td>
<td>ssh-agent</td>
<td>/tmp/8394.ssh-agent.1</td>
</tr>
<tr>
<td>8394</td>
<td>ssh-agent</td>
<td>/tmp/8394.ssh-agent.2</td>
</tr>
<tr>
<td>8394</td>
<td>ssh-agent</td>
<td>/tmp/8394.ssh-agent.3</td>
</tr>
</tbody>
</table>
(5th) Adam Bridge: NDIS Packet Scan

• Finds packets/frames in NDIS shared memory
• RAM shared between the OS and DMA NIC
• Output as text or pcap
• Decodes NetBIOS in DNS traffic
• Less chance of FPs and fake/decoys
### ndispktscan

<table>
<thead>
<tr>
<th>Offset (V)</th>
<th>Prot</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>SPort</th>
<th>DPort</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x870f5ff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>23.63.99.217</td>
<td>49311</td>
<td>80</td>
<td>ACK</td>
</tr>
<tr>
<td>0x870f6ff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>205.185.216.10</td>
<td>49286</td>
<td>80</td>
<td>ACK,RST</td>
</tr>
<tr>
<td>0x870f7ff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>208.146.36.220</td>
<td>49283</td>
<td>80</td>
<td>ACK,RST</td>
</tr>
<tr>
<td>0x870f8ff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>46.228.164.11</td>
<td>49278</td>
<td>80</td>
<td>ACK,RST</td>
</tr>
<tr>
<td>0x870f9ff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>208.146.36.221</td>
<td>49263</td>
<td>80</td>
<td>ACK,RST</td>
</tr>
<tr>
<td>0x870faff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>208.146.36.221</td>
<td>49271</td>
<td>80</td>
<td>ACK,RST</td>
</tr>
<tr>
<td>0x870fcff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>23.63.99.217</td>
<td>49311</td>
<td>80</td>
<td>ACK</td>
</tr>
<tr>
<td>0x870fdff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>23.63.99.217</td>
<td>49311</td>
<td>80</td>
<td>ACK</td>
</tr>
<tr>
<td>0x870feff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>23.63.99.217</td>
<td>49311</td>
<td>80</td>
<td>ACK</td>
</tr>
<tr>
<td>0x870ffff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>23.63.99.217</td>
<td>49311</td>
<td>80</td>
<td>ACK</td>
</tr>
<tr>
<td>0x87100ff8</td>
<td>0x06</td>
<td>192.168.203.134</td>
<td>23.63.99.217</td>
<td>49311</td>
<td>80</td>
<td>ACK</td>
</tr>
</tbody>
</table>
Monnappa Ka: Linux Memory Diff

- Compares baseline and infected Linux memory dumps
- Videos:
  - Tsunami: [https://www.youtube.com/watch?v=Fw5FrVwJslw](https://www.youtube.com/watch?v=Fw5FrVwJslw)
  - Xingiquan: [https://www.youtube.com/watch?v=_RuZ-lVysxQ](https://www.youtube.com/watch?v=_RuZ-lVysxQ)
  - Average Coder Rootkit: [https://www.youtube.com/watch?v=zKjsANieis8](https://www.youtube.com/watch?v=zKjsANieis8)
Diff analysis results:

**DIFF_PSLIST**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Pid</th>
<th>Uid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000001bca8000</td>
<td>aptd</td>
<td>2667</td>
<td>0</td>
</tr>
<tr>
<td>0x000000ae96f0</td>
<td>nm-dispatcher.a</td>
<td>2701</td>
<td>0</td>
</tr>
<tr>
<td>0x0000021e44d0</td>
<td>tsuna</td>
<td>2659</td>
<td>0</td>
</tr>
<tr>
<td>0x00001bd58000</td>
<td>strace</td>
<td>2657</td>
<td>0</td>
</tr>
</tbody>
</table>

**DIFF_PSLVIEW**

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>PID</th>
<th>plist</th>
<th>pid_hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000001bca8000</td>
<td>aptd</td>
<td>2667</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x000000ae96f0</td>
<td>nm-dispatcher.a</td>
<td>2701</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x0000021e44d0</td>
<td>tsuna</td>
<td>2659</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>0x00001bd58000</td>
<td>strace</td>
<td>2657</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

**DIFF_PIDHASHTABLE**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Pid</th>
<th>Uid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000001bca8000</td>
<td>aptd</td>
<td>2667</td>
<td>0</td>
</tr>
<tr>
<td>0x000000ae96f0</td>
<td>nm-dispatcher.a</td>
<td>2701</td>
<td>0</td>
</tr>
<tr>
<td>0x0000021e44d0</td>
<td>tsuna</td>
<td>2659</td>
<td>0</td>
</tr>
<tr>
<td>0x00001bd58000</td>
<td>strace</td>
<td>2657</td>
<td>0</td>
</tr>
</tbody>
</table>
Bartosz Inglot: Scheduled Tasks

- Finds jobs created by the `at` command and the task scheduler
- Detect persistence, lateral movement, attempts to run with SYSTEM privileges
- Based on Jamie Levy’s jobparser.py
$ python vol.py -f <FILE> schtasks

Offset(P): 0x12ab2118
ScheduledDate: 2012-11-26 19:30:00.000
MostRecentRunTime: 2012-11-26 19:30:00.021
Application: wc.exe
Author: SYSTEM
Parameters: -e -o h.out
RunInstanceCount: 1
MaxRunTime: 72:00:00.0
ExitCode: 0x00000000
Comment: Created by NetScheduleJobAdd
Joe Greenwood: HT (RCS) Attribution

• Finds Hacking Team Galileo Remote Control System (RCS)
• Not your average signature/string based detection
• Uses heuristics on named shared memory sections
  – This never touches disk
• Attribution based on "watermarks" linked to HT customers
$ python vol.py -f <FILE> attributeht
Volatility Foundation Volatility Framework 2.4
Hacking Team Galileo RCS Implant Detection - 4ARMED Ltd

<table>
<thead>
<tr>
<th>PID</th>
<th>Watermark</th>
<th>Process</th>
<th>Implant Type</th>
<th>Attrib</th>
</tr>
</thead>
<tbody>
<tr>
<td>2584</td>
<td>3OqZ1N5a</td>
<td>wscntfy.exe</td>
<td>Elite/Soldier</td>
<td>FAE-FURLAN</td>
</tr>
<tr>
<td>2584</td>
<td>3OqZ1N5</td>
<td>wscntfy.exe</td>
<td>Elite/Soldier</td>
<td></td>
</tr>
<tr>
<td>2896</td>
<td>3OqZ1N5a</td>
<td>explorer.exe</td>
<td>Elite/Soldier</td>
<td>FAE-FURLAN</td>
</tr>
<tr>
<td>2896</td>
<td>3OqZ1N5</td>
<td>explorer.exe</td>
<td>Elite/Soldier</td>
<td></td>
</tr>
</tbody>
</table>
[snip]
EG-CERT: GVol

• By May Medhat and Mohamad Shawkey
• Lightweight GUI written in Java
• Preconfigured and customizable batch scripts
• Users can add/register new plugins to support future versions of Volatility
• Integrates documentation from Art of Memory Forensics
<table>
<thead>
<tr>
<th>Batch File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Injection</td>
</tr>
<tr>
<td>Network Artifacts</td>
</tr>
<tr>
<td>Process Objects Analysis</td>
</tr>
<tr>
<td><strong>Rogue Processes</strong></td>
</tr>
<tr>
<td>Rootkits</td>
</tr>
<tr>
<td>Rootkits (Hooking)</td>
</tr>
</tbody>
</table>

### Add new batch file

Name: [ ]

[Add]

### Selected batch file plugins

<table>
<thead>
<tr>
<th>Plugin Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pslist</td>
</tr>
<tr>
<td>psscan</td>
</tr>
<tr>
<td>pstree</td>
</tr>
<tr>
<td>psxview</td>
</tr>
</tbody>
</table>

### Add Plugin to The Selected Batch File

Select Plugin: [Apihooks]

[Add]

[Done]
Alexander Tarasenko

- Pykd address space lets you run Volatility inside Windbg
Loïc Jaquemet: Haystack

• Interface between Volatility and Haystack
• Define structures and constraints manually or from C header files
• Scan for them throughout process memory
  – Alternate method to find _HEAPs
  – OpenSSL session keys
  – Any structures that Volatility scans for
haystackheap

$ vol.py -f <FILE> haystackheap -p 1668
   -r haystack.structures.win32.winxp_32.HEAP
   -c examples/winxpheap-relaxed.constraints

******************************
Pid:  1668
Record HEAP at 0x250000
Record HEAP at 0x150000
Record HEAP at 0x3f0000
**Record HEAP at 0x730000**
**Record HEAP at 0x860000**
Record HEAP at 0xba0000
Record HEAP at 0xb70000
[...]
Takeaways

• Rapidly transform memory artifact analysis into proactive use scenarios
  – Or in the very worse case, just use it

• Volatility 2.5 is out!
  – github.com/volatilityfoundation
  – volatilityfoundation.org
  – @volatility

• Congratulations to the plugin contest authors

• Get the community plugins