Troubleshooting Memory

Jamie Levy

OSDFCon 2020

$ whoami

Jamie Levy (Gleeda)

- Developer on Volatility
- Co-author: Art of Memory Forensics
- R&D at Tanium
- Lead DFIR investigator at SecureWorks, Terremark, Guidance...
- Former college professor at Queens College and John Jay College.

Plugin Contest

Will be announced later today! Keep your eyes on our Twitter account:

@volatility

Why?

- Most common issues asked about
- Thought process involved in troubleshooting

- Teaching someone how to fish ...

Types of Memory Samples

Possibilities

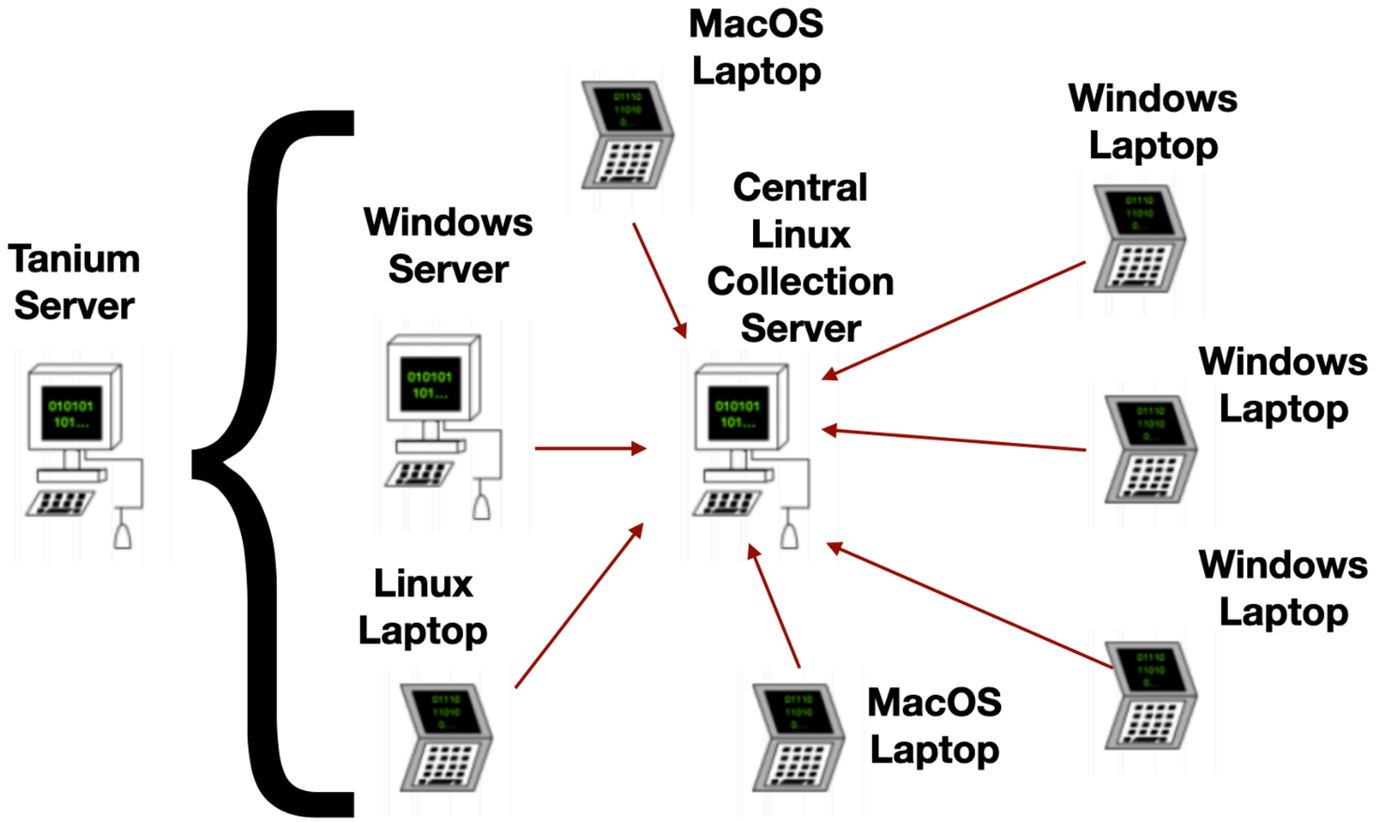
- Raw
- Hibernation Files (Windows)
  - **Windows 8 and newer not yet supported.**
- Crashdump (Windows)
- Other file formats
  - virtual box
  - vmem/vmss

Acquisition Errors

- Acquisition and Analysis can often be tricky:
  - Acquisition can fail in the middle
  - Long acquisition times may lead to “smearing”
  - Memory samples may become corrupted
  - Malware might interfere
  - Machines may update, leading to new structures

Remote Acquisition

At an Enterprise Level



What OS is it?

- Profiles* specify which operating system symbols to use in order to reconstruct memory
- The first part of analysis is figuring out which profile to use:
  - For Windows, you may use the **`kdbgscan`** plugin
    - The **`imageinfo`** plugin takes longer and doesn't show all KDBG blocks
  - For Linux, you may **`grep`** for "**`BOOT_IMAGE`**"
  - For Mac, you may use the **`mac_get_profile`** plugin

* *Profiles aren't specified in Volatility 3.0*

Check for Data

- A hex editor or other command line tools can help you determine if there is any valid data in your memory sample

```
$ <memory.img tr -d '\0' | read -n 1 || echo "Empty"
Empty
```

```
$ xxd memory2.img |grep -v "0000 0000 0000 0000 0000 0000 0000
0000" |less -I
```

Missing Dependencies?

- In order to properly process memory from some modern Windows systems, Distorm3 must be installed

```
[snip]
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not
defined)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not
defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named
distorm3)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named
distorm3)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------
------------------------------ ------------------------------
```

Acquisition Smearing

- Often the result of:
  - Buggy acquisition tools
  - Incomplete acquisition
  - Changes in memory during acquisition

```
# vol.py -f memory.dmp --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.5
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64
------------------ -------------------- ------ ------ ------ -------- ------ ------
0xfffffa8000c46b30                          0      0      0 -------- ------      0
```

Acquisition Smearing

- Depending on the situation, you may or may not have valid data

- Try “scanning” plugins
- Try “strings” utility
- Try a hex editor

Acquisition Smearing

- If you are able to get any output using “psscan”, try to use a DTB (or PDB) from a different process

```
$ python vol.py –f messed.dmp --profile=Win7SP1x86 pslist
Volatility Foundation Volatility Framework 2.5
No suitable address space mapping found
Tried to open image as:
 MachOAddressSpace: mac: need base
 LimeAddressSpace: lime: need base
 WindowsHiberFileSpace32: No base Address Space
 WindowsCrashDumpSpace64BitMap: No base Address Space
. . .
```

Acquisition Smearing

```
$ python vol.py –f messed.dmp --profile=Win7SP1x86  psscan
Volatility Foundation Volatility Framework 2.5
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ------------------------------
0x000000001da248b8 vmtoolsd.exe       1384    388 0x1ea44db0 2016-04-27 19:08:43 UTC+0000
0x000000001da285b0 svchost.exe        3256    500 0x1ea4d5c0 2016-04-27 19:11:02 UTC+0000
0x000000001da28bd0 dwm.exe             364    864 0x1ea4d3e0 2016-04-27 19:08:24 UTC+0000
0x000000001da592f8 taskhost.exe       2040    500 0x1ea4d3a0 2016-04-27 19:08:23 UTC+0000
0x000000001da75438 rundll32.exe       2240   2160 0x1ea4d420 2016-04-27 19:31:32 UTC+0000
0x000000001dad6918 conhost.exe         956    408 0x1ea4d3c0 2016-04-27 19:08:40 UTC+0000
[snip]

$ python vol.py –f messed.dmp --profile=Win7SP1x86  --dtb=0x1ea44db0  pslist
Volatility Foundation Volatility Framework 2.5
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x83db68a8 System                    4      0     89      492 ------      0 2016-04-27 19:07:50 UTC+0000
0x84fddce0 smss.exe                268      4      2       29 ------      0 2016-04-27 19:07:50 UTC+0000
0x857cd530 csrss.exe               356    348      9      427      0      0 2016-04-27 19:07:58 UTC+0000
0x857ea530 wininit.exe             396    348      3       78      0      0 2016-04-27 19:07:59 UTC+0000
0x857f1530 csrss.exe               408    388     11      272      1      0 2016-04-27 19:07:59 UTC+0000
0x85db6d40 winlogon.exe            456    388      3      115      1      0 2016-04-27 19:07:59 UTC+0000
[snip]
```

kdbgscan

Windows Memory Samples Only!

```
$ python vol.py -f ~/Desktop/Memory/memory.raw kdbgscan
[snip]
Instantiating KDBG using: Unnamed AS Win10x64_18362 (6.4.18362 64bit)
Offset (V)                    : 0xf8057863f5e0
Offset (P)                    : 0x23265e0
KdCopyDataBlock (V)           : 0xf805784bb744
[snip]
Profile suggestion (KDBGHeader): Win10x64_18362
Version64                     : 0xf805786433c8 (Major: 15, Minor: 18362)
[snip]
Build string (NtBuildLab)     : 18362.1.amd64fre.19h1_release.19
PsActiveProcessHead           : 0xfffff80578651b40 (101 processes)
PsLoadedModuleList            : 0xfffff80578661150 (201 modules)
KernelBase                    : 0xfffff80578219000 (Matches MZ: True)
[snip]
```

Using the KDBG

```
$ python vol.py -f ~/Desktop/Memory/memory.raw --kdbg=0xf805784bb744 --profile=Win10x64_18362 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------
------------------------------ ------------------------------
0xffffad08cb270040 System                    4      0    133        0 ------      0 2020-05-05 20:25:55
UTC+0000
0xffffad08cb2b0080 Registry                 88      4      4        0 ------      0 2020-05-05 20:25:47
UTC+0000
0xffffad08ce19e400 smss.exe                332      4      2        0 ------      0 2020-05-05 20:25:55
UTC+0000
0xffffad08ce87f140 csrss.exe               424    416     10        0      0      0 2020-05-05 20:26:01
UTC+0000
0xffffad08cf27c140 wininit.exe             500    416      1        0      0      0 2020-05-05 20:26:01
UTC+0000
[snip]
```

Updated Machine

- As machines are updated, the system types (vtypes) may need refreshing
- At this point you will need to create a new profile or generate new vtypes

- Sometimes reverse engineering is necessary for unknown structures

Updated Machine

```
$ python vol.py –f updated.vmem --profile=Win7SP1x86 pslist
Volatility Foundation Volatility Framework 2.5
Idle found:  0x2972740
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x83db4878 System                    4      0     77      503 ------      0 2016-08-12 21:09:27 UTC+0000
0x8514d330 smss.exe                232      4      2       29 ------      0 2016-08-12 21:09:27 UTC+0000
0x847ee568 csrss.exe               320    312      9      387      0      0 2016-08-12 21:09:34 UTC+0000
0x85950518 wininit.exe             360    312      3       76      0      0 2016-08-12 21:09:35 UTC+0000
0x8595c460 csrss.exe               372    352     10      140      1      0 2016-08-12 21:09:35 UTC+0000
0x8597e518 winlogon.exe            408    352      6      126      1      0 2016-08-12 21:09:35 UTC+0000
0x85a3bc30 services.exe            472    360     15      214      0      0 2016-08-12 21:09:36 UTC+0000
0x85a4a590 lsass.exe               480    360      8      580      0      0 2016-08-12 21:09:36 UTC+0000


$ python vol.py –f updated.vmem --profile=Win7SP1x86 psscan
Volatility Foundation Volatility Framework 2.5
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ------------------------------

$
```

Generating VTypes

- You need the kernel file (ntoskrnl.exe)
- You can use pdbparse in order to generate the vtypes

- https://github.com/moyix/pdbparse

Generating Types

```
$ python vol.py –f updated.vmem --profile=Win7SP1x86 moddump -b 0x8283b000 -D .
Volatility Foundation Volatility Framework 2.6.1
Module Base Module Name          Result
----------- -------------------- ------
0x08283b000 ntoskrnl.exe         OK: driver.8283b000.sys

$ mv driver.8283b000.sys ntoskrnl.exe
$ symchk.py -e ntoskrnl.exe
Trying http://msdl.microsoft.com/download/symbols/ntkrpamp.pdb/3A537EBBC6F945C4BE8C282C30A9A2B32/ntkrpamp.pd_
Connected. Downloading data...
0% 5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
Saved symbols to ntkrpamp.pd_
Extracting cabinet: ntkrpamp.pd_
  extracting ntkrpamp.pdb

All done, no errors.
$ pdb_tpi_vtypes.py ntkrpamp.pdb > volatility/plugins/overlays/windows/win7_sp1_x86_vtypes.py
```

Volatility 3.0

- Removes profiles
- You don’t have to generate VTypes
- It automagically generates this on the fly

- However, there can be issues where you need to manually override

Overriding in Volatility 3.0

```
$ python3 vol.py -f ~/Desktop/Memory/memory.raw windows.pslist.PsList
Volatility 3 Framework 1.2.0-beta.1
Progress:  99.22          Scanning memory_layer using PdbSignatureScanner
Unsatisfied requirement plugins.PsList.nt_symbols: Windows kernel symbols

A symbol table requirement was not fulfilled.  Please verify that:
        You have the correct symbol file for the requirement
        The symbol file is under the correct directory or zip file
        The symbol file is named appropriately or contains the correct banner

Unable to validate the plugin requirements: ['plugins.PsList.nt_symbols']
```


Overriding in Volatility 3.0

With a little help from Volatility 2

```
$ python vol.py -f ~/Desktop/Memory/memory.raw --kdbg=0xf805784bb744 --
profile=Win10x64_18362 modules|grep ntos

Volatility Foundation Volatility Framework 2.6.1

0xffffad08cb248970 ntoskrnl.exe          0xfffff80578219000           0xab7000
\SystemRoot\system32\ntoskrnl.exe


$ python vol.py -f ~/Desktop/Memory/memory.raw --kdbg=0xf805784bb744 --
profile=Win10x64_18362 moddump -b 0xfffff80578219000 -D .

Volatility Foundation Volatility Framework 2.6.1

Module Base        Module Name          Result

------------------ -------------------- ------

0xfffff80578219000 ntoskrnl.exe         OK: driver.fffff80578219000.sys
```

Overriding Volatility 3.0

```
$ mv driver.fffff80578219000.sys ntoskrnl.exe
$ symchk.py -e ntoskrnl.exe
[snip]
Trying http://msdl.microsoft.com/download/symbols/ntkrnlmp.pdb/
BE3E0FF92C7A93433D4A950A037EF6561/ntkrnlmp.pdb
Connected. Downloading data...
0% 5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80%
85% 90% 95% 100%
Saved symbols to ntkrnlmp.pdb
```

Overriding Volatility 3.0

```
$ PYTHONPATH="." python3 volatility/framework/symbols/windows/pdbconv.py -f
ntkrnlmp.pdb -o new_symbols.json


$ cat new_symbols.json |less -I
{
  "base_types": {
    "HRESULT": {
      "endian": "little",
      "kind": "int",
      "signed": false,
      "size": 4
    }, [snip]
```

Getting a Base Config File

Using a Different Working Memory Sample

```
$ python3 vol.py -f Windows\ 10\ x64-fa2e8476.vmem configwriter.ConfigWriter
Volatility 3 Framework 1.0.0-beta.1
Progress:    0.00            Scanning FileLayer using PageMapScanner
Key       Value

extra     false
primary.swap_layers true
primary.page_map_offset       1757184
primary.class       "volatility.framework.layers.intel.WindowsIntel32e"
primary.memory_layer.location “file:/Path/to/volatility3/Windows%2010%20x64-
fa2e8476.vmem"
primary.memory_layer.class    "volatility.framework.layers.physical.FileLayer"
primary.swap_layers.number_of_elements  0
```

Getting a Base Config File

```
$ cat config.json
{
  "extra": false,
  "primary.class": "volatility.framework.layers.intel.WindowsIntel32e",
  "primary.memory_layer.class":
"volatility.framework.layers.physical.FileLayer",
  "primary.memory_layer.location": “file:/Path/to/Windows%2010%20x64-
fa2e8476.vmem”,
  "primary.page_map_offset": 1757184,
  "primary.swap_layers": true,
  "primary.swap_layers.number_of_elements": 0
}
```

kdbgscan (Reminder)

(Volatility 2)

```
$ python vol.py -f ~/Desktop/Memory/memory.raw kdbgscan
[snip]
Instantiating KDBG using: Unnamed AS Win10x64_18362 (6.4.18362 64bit)
Offset (V)                    : 0xf8057863f5e0
Offset (P)                    : 0x23265e0
KdCopyDataBlock (V)           : 0xf805784bb744
[snip]
Profile suggestion (KDBGHeader): Win10x64_18362
Version64                     : 0xf805786433c8 (Major: 15, Minor: 18362)
[snip]
Build string (NtBuildLab)     : 18362.1.amd64fre.19h1_release.19
PsActiveProcessHead           : 0xfffff80578651b40 (101 processes)
PsLoadedModuleList            : 0xfffff80578661150 (201 modules)
KernelBase                    : 0xfffff80578219000 (Matches MZ: True)
[snip]
```

Getting the DTB

(Volatility 2)

```
$ python vol.py -f ~/Desktop/Memory/memory.raw --
kdbg=0xf805784bb744 --profile=Win10x64_18362 volshell

Volatility Foundation Volatility Framework 2.6.1

Current context: System @ 0xffffad08cb270040, pid=4, ppid=0
DTB=0x1ad002

>>> proc().Pcb.DirectoryTableBase.v()

1757186L
```

Overriding Volatility 3.0

Configuration file

```
$ cat config.json
{
    "primary.class": "volatility.framework.layers.intel.WindowsIntel32e",
    "primary.kernel_virtual_offset": 272702373990400,
    "primary.memory_layer.class": "volatility.framework.layers.physical.FileLayer",
    "primary.memory_layer.location": “file:///Path/to/Memory/memory.raw”,
    "primary.page_map_offset": 1757186,
    "nt_symbols.class":
"volatility.framework.symbols.windows.WindowsKernelIntermedSymbols",
“nt_symbols.isf_url”:"file:///Path/to/new_symbols.json"
}
```

Overriding Volatility 3.0

Running with a config file

```
$ python3 vol.py -c config.json windows.pslist.PsList
Volatility 3 Framework 1.0.0-beta.1

PID        PPID       ImageFileName          Offset(V)   Threads     Handles     SessionId   Wow64
CreateTime  ExitTime

4          0          System      0xad08cb270040          133         -           N/A         False
2020-05-05 20:25:55.000000         N/A
88         4          Registry    0xad08cb2b0080          4           -           N/A         False
2020-05-05 20:25:47.000000         N/A
332        4          smss.exe    0xad08ce19e400          2           -           N/A         False
2020-05-05 20:25:55.000000         N/A
424        416        csrss.exe   0xad08ce87f140          10          -           0           False
2020-05-05 20:26:01.000000         N/A
[snip]
```

Memory Sample is Still Broken

Completely Broken

- In the event that it's not completely zeroed out, go old school
  - strings utility
  - bulk_extractor
    - https://github.com/simsong/bulk_extractor
  - mftparser Volatility 2 plugin
  - hex editor

When Scanning Fails

```
$ python vol.py -f memory.raw --kdbg=0xf805784bb744 --profile=Win10x64_18362 psscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB                 Time created
Time exited
------------------ --------------- ------ ------ ------------------
---------------------------- ------------------------------
WARNING : volatility.debug    : Cannot find nt!ObGetObjectType
WARNING : volatility.debug    : Cannot find nt!ObGetObjectType
Traceback (most recent call last):
  File "vol.py", line 192, in <module>
[snip]
  File "/volatility/volatility/plugins/overlays/windows/win10.py", line 330, in TypeIndex
    return ((addr >> 8) ^ cook ^ indx) & 0xFF
TypeError: unsupported operand type(s) for ^: 'int' and 'NoneType'
```

Finding the Function

- Pull the ntoskrnl.exe from memory
- Get a clean version of it
- Examine the **`ObGetObjectType`** function
- Get the opcodes
- Search for them with **`yarascan`**
- Manually find the cookie address

```
================ B E G I N N I N G   O F   P R O C E D U R E ================


                        exp_ObGetObjectType:
00000001404ead24            lea         rax, qword [ds:rcx-0x30]
00000001404ead28            movzx       ecx, byte [ds:rcx-0x18]
00000001404ead2c            shr         rax, 0x8
00000001404ead30            movzx       eax, al
00000001404ead33            xor         rax, rcx
00000001404ead36            movzx       ecx, byte [ds:0x1403a7460]
00000001404ead3d            xor         rax, rcx
00000001404ead40            lea         rcx, qword [ds:0x1403a7900]
00000001404ead47            mov         rax, qword [ds:rcx+rax*8]
00000001404ead4b            ret
                          ; endp
```

Running Yarascan

```
$ python vol.py -f memory.raw --profile=Win10x64_18362 yarascan
-K -Y "{48 8D 41 D0 0F B6 49 E8}"
Volatility Foundation Volatility Framework 2.6.1
Rule: r1
Owner: ntoskrnl.exe
0xf805788a28c0  48 8d 41 d0 0f b6 49 e8 48 c1 e8 08 0f b6 c0 48   H.A...I.H......H
0xf805788a28d0  33 c1 0f b6 0d a7 ad ee ff 48 33 c1 48 8d 0d 9d   3.......H3.H...
[snip]
```

Getting the cookie

```
$ python vol.py -f memory.raw --kdbg=0xf805784bb744 --profile=Win10x64_18362 volshell
[snip]
>>> addr = 0xf805788a28c0
>>> for m in getmods():
...     nt_mod = m
...     break
...
>>> mode = distorm3.Decode64Bits
>>> data = nt_mod.obj_vm.read(addr, 100)
>>> ops = distorm3.Decompose(addr, data, mode, distorm3.DF_STOP_ON_RET)
>>> for op in reversed(ops):
...     if (op.size == 7 and 'FLAG_RIP_RELATIVE' in op.flags and len(op.operands) == 2 and op.operands[0].type == 'Register' and
...                                 op.operands[1].type == 'AbsoluteMemory' and
...                                 op.operands[1].size == 8):
...         addr2 = op.address + op.size + op.operands[1].disp
...
>>> cookie = obj.Object("unsigned int", offset = addr2, vm = nt_mod.obj_vm)
>>> hex(cookie)
'0x30ef575cL'
```

Success!

(psscan works without issue in Volatility 3.0)

```
$ python vol.py -f memory.raw --kdbg=0xf805784bb744 --profile=Win10x64_18362 --cookie=0x30ef575c
psscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                    PID   PPID PDB                 Time created
Time exited
------------------ --------------- ------ ------ ------------------
------------------------------ ------------------------------
0x0000ad08cb270040 System                  4      0 0x00000000001ad002 2020-05-05 20:25:55 UTC+0000
0x0000ad08cb2b0080 Registry               88      4 0x000000000eca2002 2020-05-05 20:25:47 UTC+0000
0x0000ad08cb320080 svchost.exe          1884    636 0x000000000ace9002 2020-05-05 20:26:03 UTC+0000
0x0000ad08cb322080 svchost.exe          1876    636 0x0000000013df3002 2020-05-05 20:26:03 UTC+0000
0x0000ad08cb38b040 MemCompression       1660      4 0x0000000016820002 2020-05-05 20:26:03 UTC+0000
[snip]
```

Surge: Memory Acquisition Tool

An Ounce of Prevention...

- The most recommended, stable tool for acquiring memory
- Works on all major operating systems (Windows, MacOS, Linux)
- Able to stream memory and select files across the network to a specified location
  - Includes **AWS** and **Azure**
- Uses encryption
- Available from Volexity
  - https://www.volexity.com/products-overview/surge/

Thank You!

jamie@gleeda.org

https://gleeda.org

@gleeda

@volatility

