Physical Memory Forensics for Files and Cache

Jamie Butler and Justin Murdock

BIOGRAPHY

» Jamie Butler

- Director of Research and Development at MANDIANT
- Focused on Host Analysis and Operating Systems Research

» Justin Murdock

- Student in Computer Science at RIT
- Graduating in August
- Currently a CO-OP at MANDIANT on the Product Engineering Team

AGENDA

- Traditional Forensics
- Traditional Memory Forensics
- Issues with Traditional Method
  - Missing Data
  - Misattributed Data
- Utilizing Files
  - Memory mapped files
    - Binaries
    - Data
  - Cache
- Applications
- Demos
  - MemD5 and data reduction
  - Pulling the Registry Hives from a memory image
  - Pulling a Word document from a memory image
- New Tool
- Further Work

TRADITIONAL FORENSICS

- A host has two primary components
  - Disk
  - Memory
- Memory forensics is a great way to triage a host
  - The average disk size is growing
  - Memory is relatively small comparatively
  - Intruders need to execute programs
  - An attacker often overlooks or ignores his/her memory footprint
  - Many of the artifacts the kernel uses for bookkeeping can be used for forensics

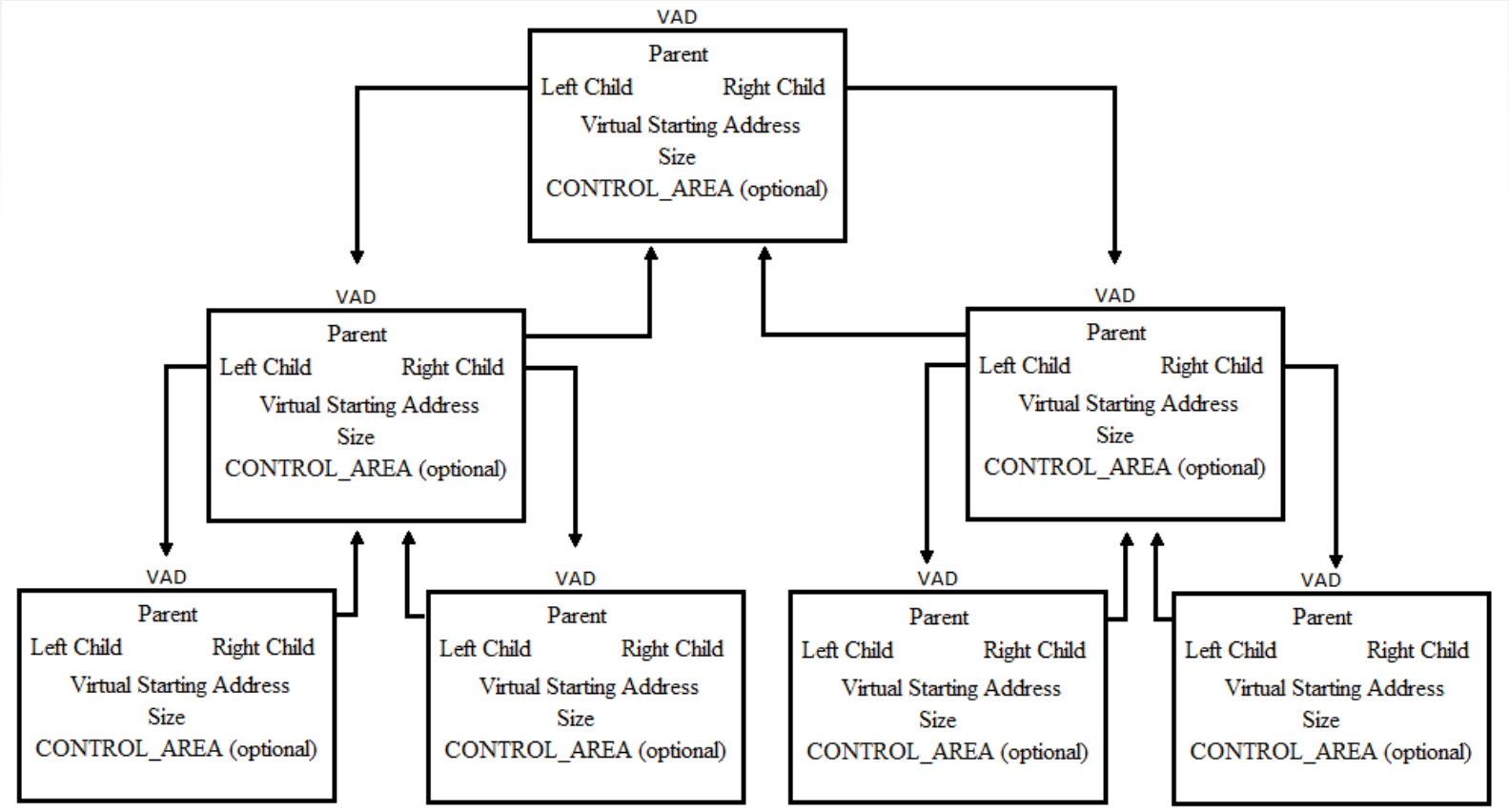
TRADITIONAL MEMORY FORENSICS

- Memory is comprised of two pieces
  - Userland memory
  - Kernel memory
- Attackers often target userland memory where processes live
  - Easier to get execution
  - More resilient to coding errors

TRADITIONAL MEMORY FORENSICS

- Process Reconstitution
  - Recover all binaries (EXEs and DLLs)
  - Focused on Virtual Address Descriptors (VADs) comprising the process' address space
    - Pointer to the parent
    - Pointer to the left child
    - Pointer to the right child
    - Virtual starting address
    - Size
    - CONTROL_AREA (optional)

VAD TREE


VAD
Parent
Left Child
Right Child
Virtual Starting Address
Size
CONTROL_AREA (optional)
VAD
Parent
Left Child
Right Child
Virtual Starting Address
Size
CONTROL_AREA (optional)
VAD
Parent
Left Child
Right Child
Virtual Starting Address
Size
CONTROL_AREA (optional)
VAD
Parent
Left Child
Right Child
Virtual Starting Address
Size
CONTROL_AREA (optional)
VAD
Parent
Left Child
Right Child
Virtual Starting Address
Size
CONTROL_AREA (optional)
VAD
Parent
Left Child
Right Child
Virtual Starting Address
Size
CONTROL_AREA (optional)
VAD
Parent
Left Child
Right Child
Virtual Starting Address
Size
CONTROL_AREA (optional)

TRADITIONAL MEMORY FORENSICS

- Scan physical memory for evidence of a process (EPROCESS block)
- Locate the Directory Table Base (DTB) in the EPROCESS for all virtual to physical address translation
- Locate the root VAD
- Enumerate the VAD tree translating each page in the virtual range to its physical location*
- Alternate approach to VADs: Use the process DTB to attempt to translate across the entire virtual address space of the operating system
  - Not practical on 64-bit systems
  - Some virtual addresses are global even if they are not used by the process

* Instead of a range, some tools use the PE header for offsets and sizes.

ISSUES WITH TRADITIONAL METHOD

- Missing Data
  - EXEs and DLLs are Memory Mapped Files
    - Shared memory across processes
    - The page table for each process is not fully populated with information about shared pages
    - Virtual addresses not in the page table fail to translate to a physical address
    - Example: Acquiring AcroRd32.exe from the Honeynet Project Challenge 3

AcroRd32.exe PID 1752	File Size in Bytes	Bytes Acquired with Traditional Approach (Ignoring FILE_OBJECT)	Bytes Acquired using FILE_OBJECT
Ntdll.dll	708,096	295,936	390,656
Wininet.dll	656,384	309,248	411,648
User32.dll	577,024	234,496	409,600
Crypt32.dll	597,504	213,504	367,104
Ace.dll	565,248	394,752	524,288

ISSUES WITH TRADITIONAL METHOD

- Misattributed Data
  - Determining what process is infected is important
  - Processes contain a lot of useful metadata
    - Username
    - Creation date
    - Location on disk
  - Data contained in the file cache appear to be part of every process when using a brute-force translation method

UTILIZING FILE OBJECTS

- File Objects represent
  - memory mapped files such as EXEs and DLLs
  - regular data files that are not mapped but contained in memory
    - Word Documents
    - PDFs
    - Registry Hives
- VADs that describe a range of memory occupied by a file contain a pointer to a Control Area
- Control Areas have pointers to the associated File Object
- File Objects contain
  - Device name such as HarddiskVolume1 (commonly C:)
  - Filename (for example: \WINDOWS\system32\kernel32.dll)
  - A pointer to three pointers depending on where the underlying data is contained
    - ImageSectionObject
    - DataSectionObject
    - SharedCacheMap

UTILIZING FILE OBJECTS

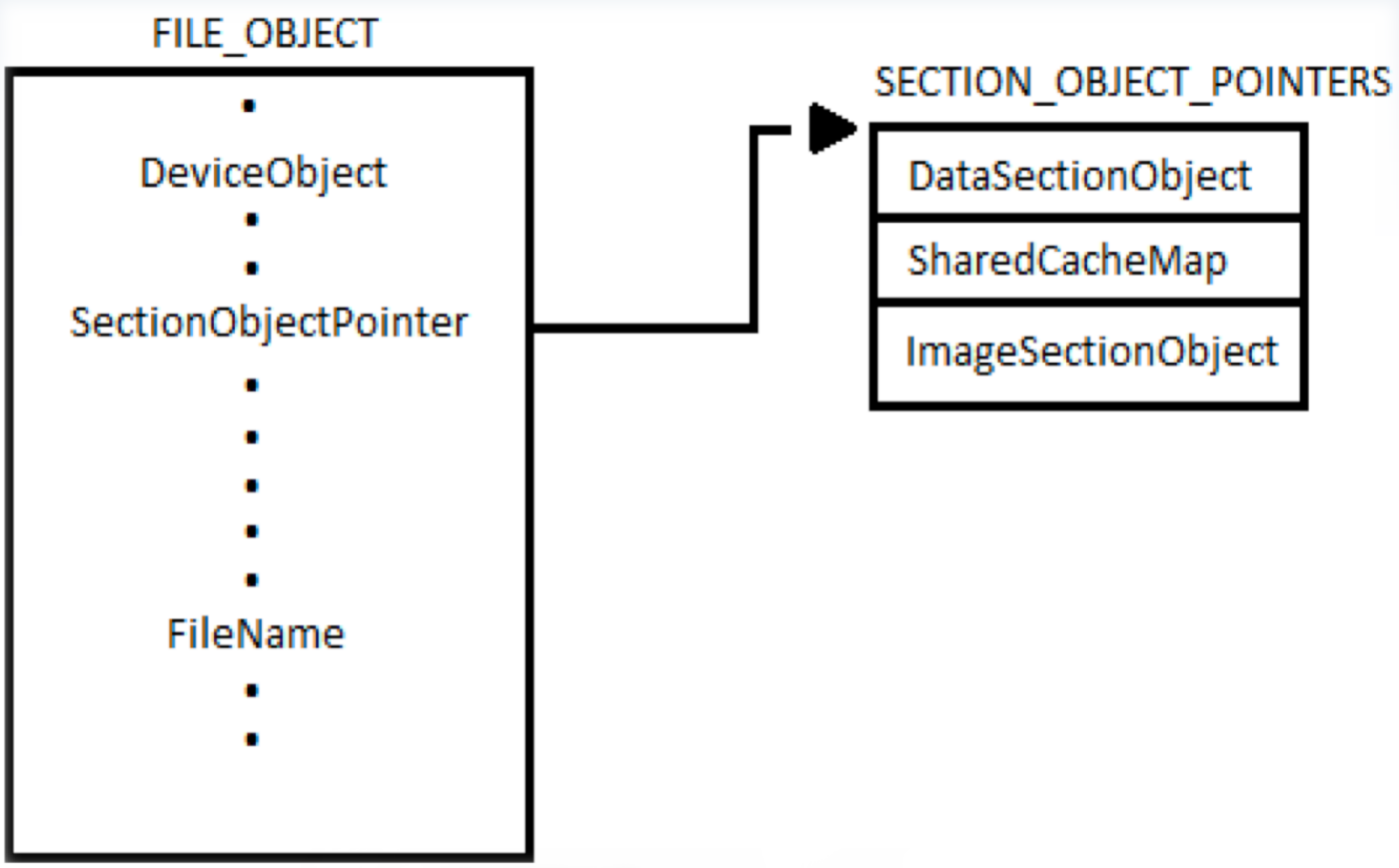


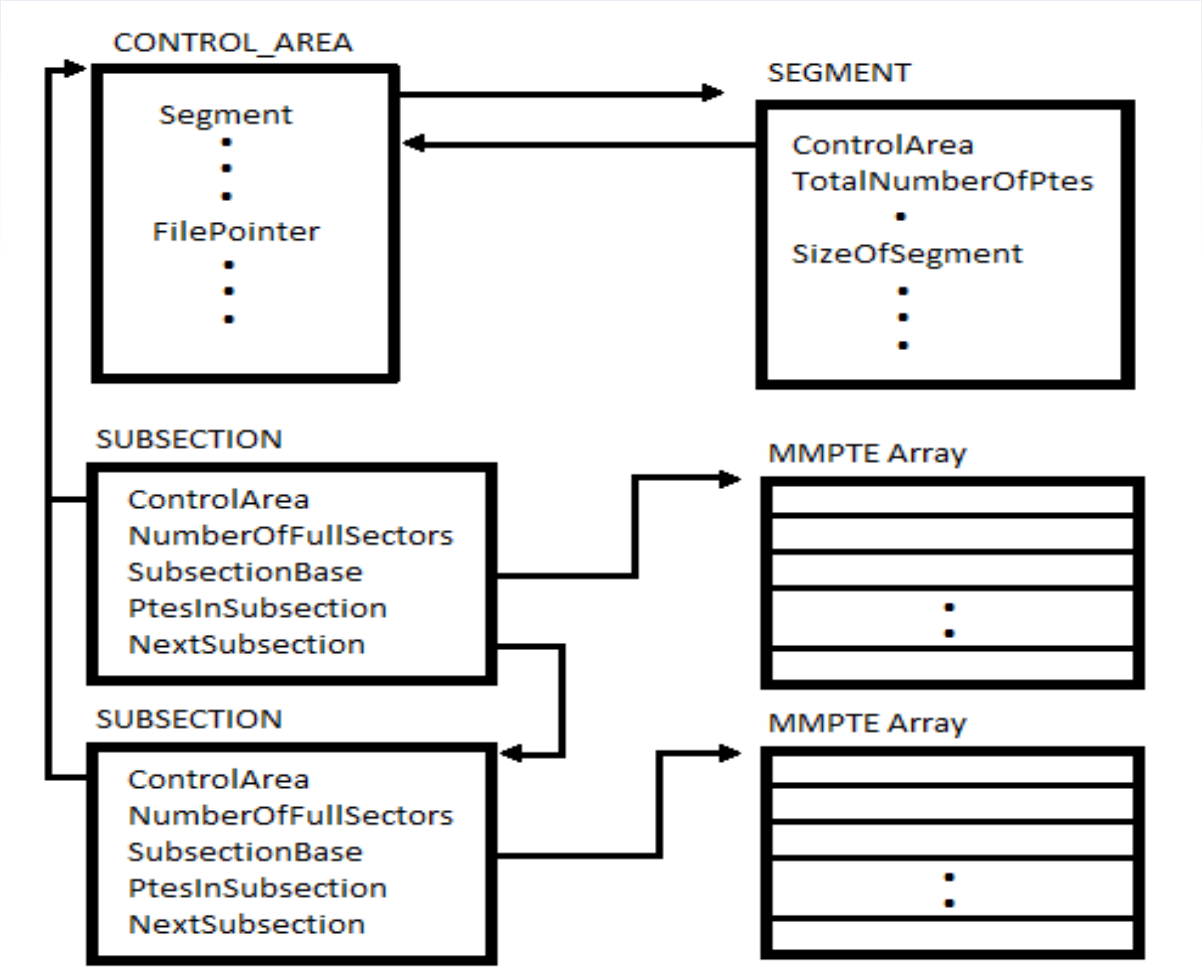
IMAGE SECTION OBJECTS

- Image Section Objects are present on binary memory mapped files
- The *ImageSectionObject* is actually a pointer to a Control Area
- The Control Area has a pointer to the Segment Object
- The Segment Objects
  - Point back to Control Area – used as a sanity check
  - Contain the size of the Segment and the total number of Prototype Page Table Entries (PTEs)
  - Segment size == Total PTEs * 0x1000 – used as a sanity check
- Subsections
  - Represent individual pieces of the file
  - Immediately after the Control Area in virtual memory
  - Point back to Control Area – used as a sanity check

IMAGE SECTION OBJECTS

- Subsections continued
  - Pointer to an array of Prototype PTEs
    - Physical address of the page in memory
    - If the Prototype PTE points back to the Subsection, that page is contained in the file on disk – not the page file(s)
    - The number of full sectors (512 bytes) and PTEs (4096 bytes) in the Subsection are vital when parsing the array of Prototype PTEs
    - By keeping track of what sector we are at within the file data, we can read the correct portion of the file from disk if the Prototype PTE indicates it is paged out
  - Pointer to the next Subsection which represents the next segment of the file

IMAGE SECTION OBJECTS



DATA SECTION OBJECTS

- Represent certain data files in memory such as Word Documents
- The Data Section Object points to a Control Area
- Almost identical to Image Section Objects
- Should have the same access performance characteristics as binaries because the content is not stored in the file cache

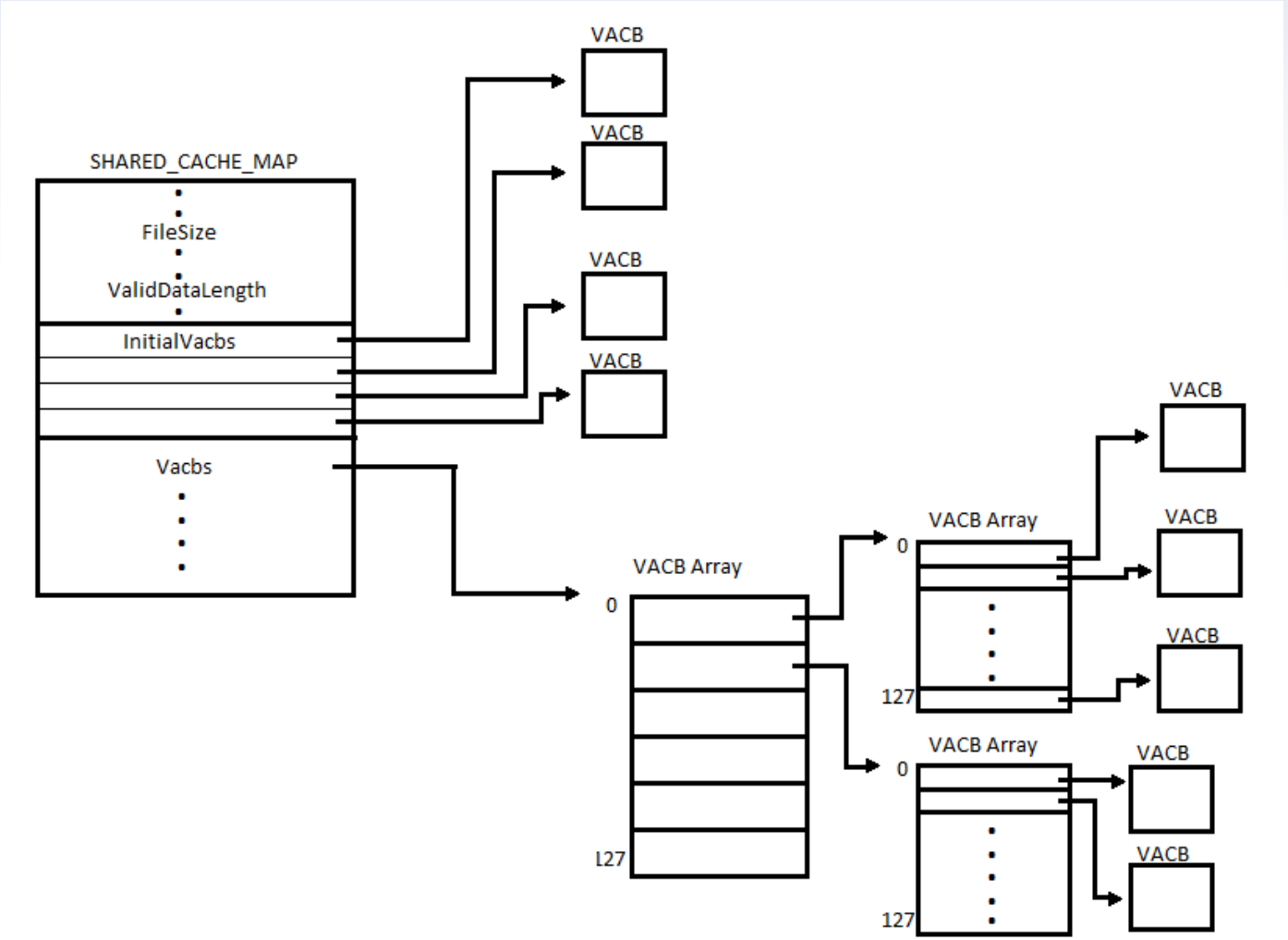
SHARED CACHE MAP

- Used to represent file data in the cache
- Contains the file size
- Contains the valid data size in the cache
- Valid data size should never be greater than file size
- Contains an embedded array of four pointers called the Initial VACB (Virtual Address Control Block) if the file in cache is 1 MB or less
- Each VACB represents 256 KB
- Contains a pointer to an array of VACB pointers for files larger than 1 MB
  - Nested structure whose depth is determined by the file size
  - Files 32 MB or less are represented by a single array (one level)
  - Larger files can reach 7 levels deep [3: 672]

$$(\lceil log\downarrow 2\ (n) \rceil - 18) / 7$$

Where n is the file size

SHARED CACHE MAP



APPLICATIONS

- File Objects can be found in
  - The process' handle table
  - The VAD tree
- Windows Registry Hives
  - Contained in the handle table of the System process
  - Found in the cache
- PDFs and other data files
  - Also found in the cache
  - Cache is completely managed by the operating system so less reliable source

APPLICATIONS

» Hashing

- Lot of data built over decades
- Can be useful when eliminating things already known (used as a filter)
  - Operating system files
  - Third-party applications
- Previously difficult to use in memory
  - Not fully using Image Section Objects
  - Other research previously tried fuzzy hashing or subsets of the file to hash against
- Using the Image Section Object during live memory analysis we can calculate an exact hash by accessing the portions that are in memory and the sections that are paged to disk – **MemD5**

APPLICATIONS

- Byte or pattern signatures
  - Classification tools such as VxClass generate byte signatures for families of malware
  - ClamAV has byte signatures for malware
  - Better binary reconstruction allows these to be used against memory the same as they would against disk

DEMOS

- Pulling the Registry Hives from a memory image
- Pulling a Word document from a memory image
- MemD5 and data reduction

UPDATED TOOL

- Memoryze 2.0
  - MemD5
  - Better process acquisition
  - Improved byte signature match
  - Can acquire
    - Registry Hives
    - Word documents
    - Files in cache
  - Improved performance
  - Available soon - http://www.mandiant.com/products/free_software/memoryze/

FURTHER WORK

- Caching across different platforms
- Address Space Layout Randomization (ASLR)
- The Security Directory in the PE file

REFERENCES

[1] Volatility. Volatile Systems <http://code.google.com/p/volatility/>.

[2] Memoryze (Memory Analyzer). <http://www.mandiant.com/products/free_software/memoryze/>.

[3] Mark E. Russinovich, and David A. Solomon. Windows Internals. 4th ed. Redmond: Microsoft, 2005. Print.

[4] Brendan Dolan-Gavitt. "The VAD Tree: A Process-eye View of Physical Memory."Digital Investigation 4 (2007): 62-64. Print.

[5] The Honeynet Project. "Challenge 3 of the Forensic Challenge 2010 - Banking Troubles The Honeynet Project." Challenge 3 of the Forensic Challenge 2010 – Banking Troubles. The Honeynet Project. Web. <http://www.honeynet.org/challenges/2010_3_banking_troubles>.

[6] Josh Smith, Matt Cote, Angelo Dell'Aera, and Nicolas Collery. "Forensic Challenge 3: Banking Troubles Solution." Honeynet. Honeynet, 12 May 2010. Web. <http://www.honeynet.org/files/Forensic_Challenge_3_-_Banking_Troubles_Solution.pdf>.

[7] Jesse Kornblum, and Kris Kendall. "Foremost." Foremost. Air Force OSI, 1 Mar. 2010. Web. <http://foremost.sourceforge.net/>.

[8] Moonsols. "Moonsols Bin2dmp.exe." MoonSols Products. Moonsols. <http://www.moonsols.com/products/>.

[9] James Okolica, and Gilbert L. Peterson. "Windows Operating Systems Agnostic Memory Analysis." Digital Investigation (2010): S48-56. Print.

[10] Seyed M. Hejazi, Mourad Debbabi, and Chamseddine Talhi. "Automated Windows Memory File Extraction for Cyber Forensics Investigation." Journal of Digital Forensic Practice (2008): 117-31. Print.

[11] Harlan Carvey. "Regripper." RegRipper. Web. <http://regripper.wordpress.com/regripper/>.

[12] Brendan Dolan-Gavitt. "Forensic Analysis of the Windows Registry in Memory.“ Digital Investigation (2008): 26-32. Print.

[13] "Finding File Contents in Memory." The NT Insider 11.1 (2004). Print.

[14] Jesse Kornblum. "Identifying Almost Identical Files Using Context Triggered Piecewise Hashing." Digital Investigation 3 (2006): 91-97. Print.

[15] R. B. Van Baar, W. Alink, and A. R. Van Ballegooij. "Forensic Memory Analysis: Files Mapped in Memory." Digital Investigation (2008): S52-57. Print.

Special thanks to Peter Silberman and Jen Andre

Questions?