Memory Forensic Analysis Essentials

Jamie Butler
Peter Silberman

Who We Are

- Jamie Butler
  - MANDIANT Intelligent Response (MIR) Agent Lead
  - Memoryze (***Memor***y + Anal***yze***) Developer
  - Co-Author *Rootkits: Subverting the Windows Kernel*
  - Director at MANDIANT

- Peter Silberman
  - MIR Agent and Memoryze Developer
  - Creator of Audit Viewer
  - Engineer at MANDIANT

MANDIANT®
INTELLIGENT INFORMATION SECURITY

Problem: Where to begin to look for a compromise?

- There is more data to look through each day
  - Hard drives are larger
  - More files exist
  - More places for an attacker to hide
- Memory analysis can facilitate quickly triaging hosts
  - If the host is still ill (has malicious software running), memory is the best place to look
  - Memory analysis can powerfully augment disk analysis

Paradigm Shift

- Changing how we focus host security
  - How many people have Anti-Virus?
  - How many people use Snort or another form of IDS?
  - Who here uses debuggers and/or disassemblers on malware?

- Apply these underlying technologies to memory
  - Anti-Virus – scan memory for process names, file handles, and mutants
  - IDS – scan memory for ports, packet fragments, strings found in malware
  - Debuggers/Disassemblers –
    - Pull binaries out of memory for analysis
    - Use binary imports and exports to identify behavior

Advantages of Memory Analysis

- Some of the best indicators of compromise are created at runtime
- Binaries are often packed or encrypted on disk
- Some malware may not even exist on disk
- Live analysis can be subverted
  - Hooking the operating system
  - Using anti-debugging tricks
  - Detecting virtualization
- Other tools may not have access to the data
  - Encrypted communications
  - Timestamps of creation and the creator/owner

Weakness of Malware Evolution in Memory

- Malware is often looking down and rarely looks up.
  - It takes a file system, registry, process view of detection
  - Hide foot print on disk
    - Polymorphism
    - Pack/Obscure binary
  - Hide persistence mechanism
  - Hide processes
- Not concerned with memory foot print
  - Open handles, sections, mutants
  - Strings and other debugging information

F.A.Q. about Memory Analysis

- Is memory analysis and acquisition possible on Windows Vista, 2003, and 2008?
- Can I analyze or acquire more than 4 GB of RAM with a 32-bit application?
- Is live memory analysis more subvertible than acquisition?
- Can the paging file(s) be used with acquired images?
- What is the difference in the memory footprint between live memory analysis and acquisition?

Using Memory Analysis to Find Evil

- File handles should be interrogated - keyloggers and other malware often have open files that look suspicious
- Mutants are easy
  - Worms and other malware create mutants to prevent re-infection
  - Randomness is often rare - alpha-numeric names look suspicious
- Strings are indicative
  - Command and control information is usually unencrypted in memory
  - Packed import tables are often built with strings
  - Debug information is sometimes present

Using Memory Analysis to Find Evil

- Context is essential
  - Processes ancestry
  - Processes with handles to other processes
  - Drivers layered on top of other devices such as a file filter driver or a keyboard driver
  - Memory sections in a process with code but no name
- Hooking is suspicious
  - Microsoft, security vendors, and attackers do it
  - Requires the analyst to understand what should be present with his default security posture

Demonstrations

- Several real life case studies that illustrate
  - Memory resident only malware
  - Injected DLLs in benign processes
  - Overcoming packers
  - Unusual handles
  - Finding suspicious hooks

Applying Memory Analysis

- Traditional forensics and reversing is not enough
- Malware has evolved to become anti-debugging and anti-virtualization
- Malware is encrypting its communication
- Memory analysis can overcome conventional shortcomings and restore to the analyst a rich dataset to use in identification
- Use memory analysis to identify suspicious processes and drivers by characterizing behavior
  - DLLs
  - Imported functions
  - Layering
  - Hooked functions
  - Files
  - Strings
  - Ports
  - Mutex

Resources

- Tools
  - Memoryze - http://www.mandiant.com/software/memoryze.htm
  - Audit Viewer - http://www.mandiant.com/software/mav.htm
- Email
  - james.butler@mandiant.com
  - peter.silberman@mandiant.com
- Blog
  - http://blog.mandiant.com

MANDIANT
INTELLIGENT INFORMATION SECURITY