Memory Forensics
Analysis Poster

Memory analysis is the decisive victory on the battlefield between offense and defense, giving the upper hand to incident responders by exposing injection and hooking techniques that would otherwise remain undetected.

Memory Analysis will prepare your defenses:
• Discover zero-day malware
• Detect compromises
• Uncover evidence that others miss

Counters to Memory Forensics: Modern Anti-Analysis Techniques

Subverting Memory Acquisition

Dementia by Jake Williams

An impressive advancement in "anti-analysis" research was presented by Luka Milkovic at the 29th Chaos Communication Congress in December 2012. The tool creates a process with a specific name and ID, but does not actually run. This allows the process to be analyzed and monitored by the defender, while the attacker remains undetected.

Anti-Analysis: Spinning the Wheels of the Forensic Examiner

Address Space Layout Randomization (ASLR) and StackGuard prevent memory analysis tools from directly interacting with the victim's memory. Tools that rely on process ID injection, like PT Stones, are circumvented. This leaves tools like Volatility and Rekall with minimal capabilities.

Evasive of Malicious Code Detection Techniques

Gargoyle by Josh Campbell

One of the methods we use to identify code execution is to look for repeated memory locations that are not mapped to disk. Gargoyle implements a technique called "threadless" that can hide code execution by replacing the memory page with an equivalent Page fault. After Gargoyle executes, it again calls VirtualProtectEx to return to RWX protections, thus hiding any evidence.

Memory Forensics

Recover Memory-Resident Evidence of Execution: Shimcohemech

By Fred House, Andrew Davis, and Claudia Teodoru

The use of shimcache artifacts in many investigations has been limited because data is not updated in the registers until the system is shut down. A winning submission to the 2015 Volatility plug-in contest, this researcher authored a parser plugin that extracts these entries from the Application Compatibility Cache database in Windows memory. Despite changes in structure and the method of organization of these entries across versions of Windows, shimcacheViewer now supports versions from WinXPSP2 to Windows2012R2.

Decompress Win+HiberSift and Carve Hibernation Slack: Hibernation Recon

Hibernation Recon by Arual Sam

Decompressing the Win+HiberSift tool is a critical step in analyzing physical memory for a hibernating machine. Our tools at the time could not decompress, let alone carve, the hibernation slack memory file that was the aforementioned shimcacheViewer. An impressive advancement in "anti-analysis" research was presented by Luka Milkovic at the 29th Chaos Communication Congress in December 2012. His tool creates a process with a specific name and ID, but does not actually run. This allows the process to be analyzed and monitored by the defender, while the attacker remains undetected.

Threat Detection

Advances in Memory Forensics

Recover Memory-Resident Evidence of Execution: Shimcohemech

By Fred House, Andrew Davis, and Claudia Teodoru

The use of shimcache artifacts in many investigations has been limited because data is not updated in the registers until the system is shut down. A winning submission to the 2015 Volatility plug-in contest, this researcher authored a parser plugin that extracts these entries from the Application Compatibility Cache database in Windows memory. Despite changes in structure and the method of organization of these entries across versions of Windows, shimcacheViewer now supports versions from WinXPSP2 to Windows2012R2.

Decompress Win+HiberSift and Carve Hibernation Slack: Hibernation Recon

Hibernation Recon by Arual Sam

Decompressing the Win+HiberSift tool is a critical step in analyzing physical memory for a hibernating machine. Our tools at the time could not decompress, let alone carve, the hibernation slack memory file that was the aforementioned shimcacheViewer. An impressive advancement in "anti-analysis" research was presented by Luka Milkovic at the 29th Chaos Communication Congress in December 2012. His tool creates a process with a specific name and ID, but does not actually run. This allows the process to be analyzed and monitored by the defender, while the attacker remains undetected.

Threat Detection

Advances in Memory Forensics

Recover Memory-Resident Evidence of Execution: Shimcohemech

By Fred House, Andrew Davis, and Claudia Teodoru

The use of shimcache artifacts in many investigations has been limited because data is not updated in the registers until the system is shut down. A winning submission to the 2015 Volatility plug-in contest, this researcher authored a parser plugin that extracts these entries from the Application Compatibility Cache database in Windows memory. Despite changes in structure and the method of organization of these entries across versions of Windows, shimcacheViewer now supports versions from WinXPSP2 to Windows2012R2.

Decompress Win+HiberSift and Carve Hibernation Slack: Hibernation Recon

Hibernation Recon by Arual Sam

Decompressing the Win+HiberSift tool is a critical step in analyzing physical memory for a hibernating machine. Our tools at the time could not decompress, let alone carve, the hibernation slack memory file that was the aforementioned shimcacheViewer. An impressive advancement in "anti-analysis" research was presented by Luka Milkovic at the 29th Chaos Communication Congress in December 2012. His tool creates a process with a specific name and ID, but does not actually run. This allows the process to be analyzed and monitored by the defender, while the attacker remains undetected.

Threat Detection
What Lies Within: Windows Memory Analysis

We are in a cybersecurity arms race as incident responders, faced with a growing sophistication of threats, posed by actors both internal and external to our environment. Our ability to effectively and efficiently detect and contain malicious actors inside our environment hinges on visibility into the current system state of our endpoint. The details uncovered through memory forensics allow us to baseline normal functions and spot significant anomalies indicative of malicious activity. This poster provides insights into the most relevant Windows internal structures for forensic analysis. Though there are far more members of each structure than shown here, these are the most pertinent for spotting malicious activity and subversion.

Security Protections

Kernel Patch Protection (aka PatchGuard)

Modern x64 Windows implements a functionality called Kernel Patch Protection (sometimes referred to as PatchGuard), which restricts key system structures, including (but not limited to) the doubly-linked lists that track most objects on Windows. Many researchers have relied on kernel modules in the past, but they require new level access to the system. Rootkits, packet sniffers, and keyloggers may be able to create this level of access. The module structure of the _LDR_DATA_TABLE_ENTRY structure contains a list of DLLs that are initialized but not used at this time, which allows for a straightforward identification of other processes. The _LDR_DATA_TABLE_ENTRY structure contains a list of DLLs that are initialized but not used at this time, which allows for a straightforward identification of other processes. The _LDR_DATA_TABLE_ENTRY structure contains a list of DLLs that are initialized but not used at this time, which allows for a straightforward identification of other processes.

ObjectTable

The Windows 6.1 structure block of recently unloaded kernel modules (device drivers). This is useful for finding resources (or misleading legitimate device drivers).

ObjectTable

The windows 6.1 structure block of recently unloaded kernel modules (device drivers). This is useful for finding resources (or misleading legitimate device drivers).

3) VAD

VADs (Virtual Address Descriptions) are used by the memory manager to track ALL memory allocated by the system. Many and rootkits may create a VAD that contains a memory region, but hiding from the memory manager is easy. It can’t be any memory, it will give away!

4) _EPROCESS

The _EPROCESS is perhaps the most important structure in memory forensics. The _EPROCESS contains more than 100 members, many of them pointers to other structures. The _EPROCESS gives us the PID and parent PID of a given process. Analyzing PID relationships between processes can reveal much. For more information, see the SAM钢材 blog post “Kernel Filesystem”.

5) Process Environment Block (PEB)

This structure contains pointers to three linked lists of loaded modules in a given process. Each is ordered differently (order by launch, order of initialization, and order by memory addresses). Sometimes rootkits will inject a DLL into an legitimate Windows service, then try to hide. Rather than rely on the DLL order, they may use the memory order.

6) _LDR_DATA_TABLE_ENTRY

- _LDR_DATA_TABLE_ENTRY
- UserMode - The base address of the DLL
- ExitPoint - Entry point of the DLL
- OffsetToImage - Size of the DLL in memory
- FullDllName - Full path name of the DLL
- TimeDateStamp - The complete time stamp for the DLL

7) ThreadListHead

- ThreadListHead - List of active threads (LDRriad)

8) Process Struct (EPROCESS)

- _EPROCESS
- Pointer to the root of the _EPROCESS

9) Process Environment Block (PEB)

- _PEB
- Pointer to the root of the _PEB

10) Unloaded Modules

The Windows 6.1 structure block of recently unloaded kernel modules (device drivers). This is useful for finding resources (or misleading legitimate device drivers).

11) Process Environment Block (PEB)

The _PEB contains pointers to the _PEB_LDR_DATA_data structure (discussed below). It also contains a flag that indicates whether the process is in kernel mode.

12) Process Environment Block (PEB)

Sometimes malware will inject a DLL into a legitimate Windows service, then try to hide. But they'd probably forget that the DLL order list is ordered differently (order of memory addresses).

13) Process Environment Block (PEB)

- _PEB
- Pointer to the root of the _PEB

14) Process Environment Block (PEB)

- _PEB
- Pointer to the root of the _PEB

15) Process Environment Block (PEB)

- _PEB
- Pointer to the root of the _PEB

FOR526: Memory Forensics In-Depth

AUTHORS:
Aldisa Torres @alsertorres
Jake Williams @malwareoakj

In today’s enterprise investigations, memory forensics plays a crucial role. Threats are evolving, and attacks are becoming increasingly sophisticated. Skilled cybercriminals are leveraging advanced techniques to evade detection, making the analysis of memory contents crucial. Memory forensics allows us to analyze the system’s memory, discovering malicious activities and finding evidence of recent or ongoing attacks. This poster provides a comprehensive overview of memory forensics, focusing on the key structures and techniques used in Windows memory analysis.

1) PlsLoadedModuleList

The PlsLoadedModuleList structure of the KBDK points to the list of loaded kernel modules (device drivers). Many rootkits and malware rely on kernel modules because they require new level access to the system. Rootkits, packet sniffers, and keyloggers may be able to create this level of access. The module structure of the _LDR_DATA_TABLE_ENTRY structure contains a list of DLLs that are initialized but not used at this time, which allows for a straightforward identification of other processes. The _LDR_DATA_TABLE_ENTRY structure contains a list of DLLs that are initialized but not used at this time, which allows for a straightforward identification of other processes.

2) Process Environment Block (PEB)

The _PEB contains pointers to the _PEB_LDR_DATA_data structure (discussed below). It also contains a flag that indicates whether the process is in kernel mode.

3) VAD

Virtual Address Descriptions (VADs) are used by the memory manager to track all memory allocated by the system. Many rootkits and malware may use a VAD that contains a memory region, but this is easy to detect. It can’t be any memory, it will give away!

4) _EPROCESS

The _EPROCESS is perhaps the most important structure in memory forensics. The _EPROCESS contains more than 100 members, many of them pointers to other structures. The _EPROCESS gives us the PID and parent PID of a given process. Analyzing PID relationships between processes can reveal much. For more information, see the SAM钢材 blog post “Kernel Filesystem”.

5) Process Environment Block (PEB)

This structure contains pointers to three linked lists of loaded modules in a given process. Each is ordered differently (order by launch, order of initialization, and order by memory addresses). Sometimes malware will inject a DLL into a legitimate Windows service, then try to hide. But they’d probably forget that the DLL order list is ordered differently (order of memory addresses).

6) _LDR_DATA_TABLE_ENTRY

- _LDR_DATA_TABLE_ENTRY
- UserMode - The base address of the DLL
- ExitPoint - Entry point of the DLL
- OffsetToImage - Size of the DLL in memory
- FullDllName - Full path name of the DLL
- TimeDateStamp - The complete time stamp for the DLL

Note that many internal OS structures are doubly-linked lists. The pointers in the lists actually point to the pointer in the next structure. However, for clarity of illustration, we have chosen to show the type of structure they point to. Also, note that the PlsActiveProcessLinks member of the KBDK structure points to the ActiveProcessLinks member of the _EPROCESS structure. For clarity, we depict the pointer pointing to the base of the PlsActiveProcessLinks member of the _EPROCESS structure. We feel that this depiction illustrates this more clearly.