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 team to:
- Discover zero-day malware
- Detect compromises
- Uncover evidence that others miss

Rekall Memory Forensic Framework

The Rekall Memory Forensic Framework is a collection of memory acquisition and analysis tools implemented in Python and designed for memory acquisition and analysis. It is a free and open-source project for memory forensic researchers. For more information on this tool, visit rekal-forensic.com.

Extracting Process Details

Emitting Process Details

- Process Details
  - PID
  - Name
  - Image path
  - cmd

- Module Details
  - Name
  - Offset
  - Size

Tip: To find process IDs, use the "tasklist" command in Windows.

Detecting Code Injection

- Use the "dumpall" command to dump all memory sections.

- Use the "dumpall" command to dump all memory sections.

- Use the "dumpall" command to dump all memory sections.

Tip: To find process IDs, use the "tasklist" command in Windows.

Counters to Memory Forensics: Modern Anti-Analysis Techniques

Subverting Memory Acquisition

- Dementia by Jake Williams

- Anti-analysis methods: "Dementia" research was presented by Jake Williams at the 26th Chaos Communication Congress in December 2012. He built a Dementia, centroid memory capture by intercepting the MIR of a key through the use of online heuristics and a full system malware filter. The output of a memory acquisition tool is manipulated so that any reference to the target process and its kernel objects is removed and the unchanged memory image is less suspicious for the memory forensics process.

For more on this, visit: https://events.ccc.de/congress/2012/Fahrplan/attachments/2231_Defeating%20Windows%20memory%20forensics.ppt

Anti-Analysis: Spinning the Wheels of the Forensic Examiner

- frozencore (Formerly FOR408)

- frozencore (Formerly FOR408)

- frozencore (Formerly FOR408)

Anti-analysis methods: "frozencore" research was presented by Jake Williams at the 26th Chaos Communication Congress in December 2012. He built a frozencore, centroid memory capture by intercepting the MIR of a key through the use of online heuristics and a full system malware filter. The output of a memory acquisition tool is manipulated so that any reference to the target process and its kernel objects is removed and the unchanged memory image is less suspicious for the memory forensics process.

For more on this, visit: https://events.ccc.de/congress/2012/Fahrplan/attachments/2231_Defeating%20Windows%20memory%20forensics.ppt

Evasive of Malicious Code Detection Techniques

Gargoyle by Josh Compton

- One of the methods we use to identify code injection (see Step 4 above) is to look for readable memory that is not mapped to disk. Gargoyle implements a creative model of covert persistence technique, and it was used to create our tool, only memory that is unmapped to any file and an asynchronous procedure call hooked at a time that a DPMF module to invoke Rekall Plugins to change permissions to R/W. After Gargoyle has executed, 4-times, RealPivot returns to R/W permission to its normal model.

For more on this, visit: https://github.com/joshcompton/gargoyle
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 endpoints. The details uncovered through memory examination allows us to baseline normal functions and spot significant anomalies indicative of malicious activity. This poster provides insight 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.

Securities Protections

Kernel Patch Protection (aka PatchGuard)

Windows 8 introduces a functionality called Kernel Patch Protection (renamed to SecureGuard). KPP checks key system structures, including (but not limited to) the doubly-linked lists that track most objects on Windows. In particular, KPP makes the DLMODE technique of unloading a process from the process list disallowed. When KPP detects an unauthorized modification, it causes a BSOD to halt the system. As a result, Windows kernel modules now use kernel callbacks, Asynchronous Procedure Calls (APC), and Deferred Procedure Calls (DPC) to load instead of this old “launch a process and use DKOM to hide it” technique.

Kernel Object Obfuscation

Just as we do in memory forensics, many rootkits have relied on the KDBG to locate key operating system structures. As of Windows 8, however, the KDBG is encrypted to prevent rootkits from easily locating it. This does not impact operations since the KDBG is not used under normal system operation. If the system crashes, the Kernel Crash Stack decodes the KDBG before storing the crash dump data in the page file. From the Idle process DTB (directory table base), all other required structures can be determined. From the Idle process (the Idle process is really just an accounting structure), then uses this to find the image base of the kernel. Then, the KDBG (if needed at all) can be found deterministically, rather than using the scanning approach to find the KDBG used by Volatility. From the Idle process DTB, all other required structures can be identified.

Unloaded Modules

The Windows 8 crash dump has a list of recently unloaded kernel modules (device drivers). This is useful for finding unloaded (and misbehaving legitimate device drivers).

3) VAD

VADs (Virtual Address Descriptors) are used by the memory manager to track ALL memory allocated to a process. Many malware variants use kernel modules because they require low level access across the system. Rootkits, packet sniffers, and many keyloggers are simply load in the standard VAD list. The members of the list are _VAD, _VAD, _VADVH structure. The _VAD structure can be found by searching for structures that look like a tree. The VADs in memory allow us to see what the malware is using.

4) _EPROCESS

The _EPROCESS is perhaps the most important structure in memory forensics. The _EPROCESS contains a list of loaded kernel modules (device drivers). This is useful for finding loaded (and misbehaving legitimate device drivers).

5) Process Environment Block

The _PEB contains pointers to the _PEB_LDR_DATA structure (discussed below). It also contains a flag that indicates if a debugger is attached to a process. Forensic examiners will often check a process for the presence of a debugger as an unmitigating factor. Finally, this PE contains a pointer to the command line arguments that were supplied the process on creation.

6) LDR_DATA_TABLE_ENTRY

Where are the thread list structures in the poster? Sorry, we just don’t have room to do them justice. For example, instead of showing the thread list, we instead show the calling thread environment. We are not going to go into overdrive with this, but we also have the effect of inhibiting some scanning plugins.

FOR526: Memory Forensics In-Depth

AUTHORS: Alizza Torres, @sibertas, and Jake Williams, @malwarejake

In today’s enterprise investigations, memory forensics plays a crucial role in unraveling the details of what happened on the system. Recent large-scale malware infections have involved attackers implementing advanced anti-analysis techniques, making the system memory the battleground between offense and defense. Skilled incident responders use memory forensics skills to reveal “ground truth” of malicious activity and move more swiftly to remediation.

Learn more about FOR526: Memory Forensics In-Depth at www.sans.org/FOR526

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 PsActiveProcesschk member of the _KDBG structure points to ActiveProcesschkList, a doubly-linked list of _EPROCESS structures. For clarity, we depict the pointer pointing to the base of the _EPROCESS structure. We feel that this depiction illustrates this more clearly.

---

1) PsLoadedModuleList

The PsLoadedModuleList structure of the KDBG points to the list of loaded kernel modules (device drivers). Many malware variants use kernel modules because they require low-level access across the system. Rootkits, packet sniffers, and many keyloggers are simply load in the standard VAD list. The members of the list are _LDR_DATA_TABLE_ENTRY structure. The _LDR_DATA_TABLE_ENTRY structure can be found by searching for structures that look like a tree. The VADs in memory allow us to see what the malware is using.

2) Unloaded Modules

The Windows 8 crash dump has a list of recently unloaded kernel modules (device drivers). This is useful for finding unloaded (and misbehaving legitimate device drivers).

3) VAD

VADs (Virtual Address Descriptors) are used by the memory manager to track ALL memory allocated to a process. Many malware variants use kernel modules because they require low level access across the system. Rootkits, packet sniffers, and many keyloggers are simply load in the standard VAD list. The members of the list are _VAD, _VAD, _VADVH structure. The _VAD structure can be found by searching for structures that look like a tree. The VADs in memory allow us to see what the malware is using.

4) _EPROCESS

The _EPROCESS is perhaps the most important structure in memory forensics. The _EPROCESS contains a list of loaded kernel modules (device drivers). This is useful for finding loaded (and misbehaving legitimate device drivers).

5) Process Environment Block

The _PEB contains pointers to the _PEB_LDR_DATA structure (discussed below). It also contains a flag that indicates if a debugger is attached to a process. Forensic examiners will often check a process for the presence of a debugger as an unmitigating factor. Finally, this PE contains a pointer to the command line arguments that were supplied the process on creation.

6) LDR_DATA_TABLE_ENTRY

Where are the thread list structures in the poster? Sorry, we just don’t have room to do them justice. For example, instead of showing the thread list, we instead show the calling thread environment. We are not going to go into overdrive with this, but we also have the effect of inhibiting some scanning plugins.

FOR526: Memory Forensics In-Depth

AUTHORS: Alizza Torres, @sibertas, and Jake Williams, @malwarejake

In today’s enterprise investigations, memory forensics plays a crucial role in unraveling the details of what happened on the system. Recent large-scale malware infections have involved attackers implementing advanced anti-analysis techniques, making the system memory the battleground between offense and defense. Skilled incident responders use memory forensics skills to reveal “ground truth” of malicious activity and move more swiftly to remediation.

Learn more about FOR526: Memory Forensics In-Depth at www.sans.org/FOR526

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 PsActiveProcesschk member of the _KDBG structure points to ActiveProcesschkList, a doubly-linked list of _EPROCESS structures. For clarity, we depict the pointer pointing to the base of the _EPROCESS structure. We feel that this depiction illustrates this more clearly.

---

1) PsLoadedModuleList

The PsLoadedModuleList structure of the KDBG points to the list of loaded kernel modules (device drivers). Many malware variants use kernel modules because they require low-level access across the system. Rootkits, packet sniffers, and many keyloggers are simply load in the standard VAD list. The members of the list are _LDR_DATA_TABLE_ENTRY structure. The _LDR_DATA_TABLE_ENTRY structure can be found by searching for structures that look like a tree. The VADs in memory allow us to see what the malware is using.

2) Unloaded Modules

The Windows 8 crash dump has a list of recently unloaded kernel modules (device drivers). This is useful for finding unloaded (and misbehaving legitimate device drivers).

3) VAD

VADs (Virtual Address Descriptors) are used by the memory manager to track ALL memory allocated to a process. Many malware variants use kernel modules because they require low level access across the system. Rootkits, packet sniffers, and many keyloggers are simply load in the standard VAD list. The members of the list are _VAD, _VAD, _VADVH structure. The _VAD structure can be found by searching for structures that look like a tree. The VADs in memory allow us to see what the malware is using.

4) _EPROCESS

The _EPROCESS is perhaps the most important structure in memory forensics. The _EPROCESS contains a list of loaded kernel modules (device drivers). This is useful for finding loaded (and misbehaving legitimate device drivers).

5) Process Environment Block

The _PEB contains pointers to the _PEB_LDR_DATA structure (discussed below). It also contains a flag that indicates if a debugger is attached to a process. Forensic examiners will often check a process for the presence of a debugger as an unmitigating factor. Finally, this PE contains a pointer to the command line arguments that were supplied the process on creation.

6) LDR_DATA_TABLE_ENTRY

Where are the thread list structures in the poster? Sorry, we just don’t have room to do them justice. For example, instead of showing the thread list, we instead show the calling thread environment. We are not going to go into overdrive with this, but we also have the effect of inhibiting some scanning plugins.

FOR526: Memory Forensics In-Depth

AUTHORS: Alizza Torres, @sibertas, and Jake Williams, @malwarejake

In today’s enterprise investigations, memory forensics plays a crucial role in unraveling the details of what happened on the system. Recent large-scale malware infections have involved attackers implementing advanced anti-analysis techniques, making the system memory the battleground between offense and defense. Skilled incident responders use memory forensics skills to reveal “ground truth” of malicious activity and move more swiftly to remediation.

Learn more about FOR526: Memory Forensics In-Depth at www.sans.org/FOR526

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 PsActiveProcesschk member of the _KDBG structure points to ActiveProcesschkList, a doubly-linked list of _EPROCESS structures. For clarity, we depict the pointer pointing to the base of the _EPROCESS structure. We feel that this depiction illustrates this more clearly.