Quality of Inconsistencies in (Windows) Memory Dumps

Lisa Rzepka
Universität der Bundeswehr München
FI CODE
lisa.rzepka@unibw.de

Harald Baier
Universität der Bundeswehr München
FI CODE
harald.baier@unibw.de

Abstract. Acquiring main memory is common during forensic investigations, as it typically contains valuable information which is hardly available by other methods, e.g., information about executed processes, running network connections or encryption keys. In practice the main memory is mostly obtained using kernel-level software tools which run concurrently to the system. This introduces a phenomenon called page smearing, i.e., content mismatches or *inconsistencies*, which may influence the subsequent forensic analysis of the acquired memory dumps. In order to measure the impact of inconsistencies on the analysis, suitable indicators and quality assessments are needed. This work presents the state of the art regarding inconsistency considerations and points to future directions.

1 Introduction

Random-access memory (RAM) might contain information which is not obtainable from other sources, for example, encryption keys or running network connections. Therefore, it is common practice to acquire the content of RAM in a file which is called a memory dump. This copy of memory is typically retrieved on a running system, as RAM content disappears after powering it off. The concurrency of the running system and the kernel-level memory acquisition tool causes inconsistencies in the obtained memory dumps, i.e. content mismatches or page smearing [1], as the main memory is acquired sequentially and the memory pages may change during this process.

The occurrence of such inconsistencies appears to be normal in Linux and Windows operating systems [3, 7]. For this reason, the impact of inconsistencies on the subsequent analysis of memory dumps, i.e., their quality, needs to be researched. In this work we present the state of the art of inconsistency-induced quality considerations of RAM and point to further research directions.

2 Inconsistencies: The Story so far

In this section we present two of our research papers which address the quantification of inconsistencies and a first attempt to measure the quality of memory dumps using two inconsistency indicators.

2.1 Quantitative Assessment of Inconsistencies in Windows

Rzepka et al. [7] conduct a quantitative assessment of inconsistencies present in Windows memory dumps by using two inconsistency indicators: 1) causal inconsistencies within self-injected memory data structures as introduced by Ottmann et al. [3] and 2) inconsistencies in memory management kernel data structures, namely the virtual address descriptor (VAD) tree.

Causal consistency is defined by Ottmann et al. [3] as follows: In two memory regions `r1` and `r2` the two events `e1` and `e2` are happening, respectively. The event `e1` is the cause of event `e2`. If both events, and in particular the cause of event e2, are in the memory snapshot generated by the memory acquisition tool, the snapshot is called *causally consistent*. This notion of consistency is measured in [3] by using a dedicated software called *pivot program*.

Rzepka et al. [7] propose and implement a Volatility3[1] [2] plugin to measure VAD inconsistencies. Their plugin counts all nodes in the VAD tree while traversing it and compares this number to a variable found in the kernel symbol table called `vadCount`. If the two numbers differ, an inconsistency is found.

In order to apply both approaches, a dataset is needed. During their evaluation Rzepka et al. [7] investigate the impact of different influencing factors on the number of inconsistencies, in particular, system workload (defined by an activity level), execution time of the memory acquisition tool WinPmem[2] and number of threads of the pivot program. The analysis is based on more than 180 memory dumps, which Rzepka et al. [7] generate in an automated approach. As a result, the system workload seems to have a stronger impact on the execution time than the number of threads. This supports the assumption that a higher workload results in a lower CPU share of WinPmem per time frame, which leads to a longer acquisition execution time. Additionally, both activity level and execution time influence the number of VAD inconsistencies. Most of the inconsistencies are found in background processes of the Windows operating system (e.g. services). Regarding causal inconsistencies, the influence of the higher number of threads is only visible when using at least 8 threads for the pivot program. This

[1] https://github.com/volatilityfoundation/volatility3
[2] https://github.com/Velocidex/WinPmem

supports the assumption that the pages of the pivot program change more frequently due to the higher amount of threads which in turn leads to a higher amount of causal inconsistencies.

2.2 Towards Quality

As a step towards a reliable metric for assessing the quality of memory dumps, Rzepka et al. [6] present a scenario-based evaluation method to test memory acquisition tools regarding their ability to retrieve certain forensic artifacts. In their work, Rzepka et al. [6] generate a dataset consisting of 1600 Windows memory dumps in an automated way using ForTrace++[3] [9]. The authors focus on four forensically relevant scenarios, in particular, the ability of the analysis tool Volatility to retrieve artifacts of a running process (scenario 1), an opened network connection (scenario 2), an encryption key of a VeraCrypt container (scenario 3) and an opened image file (scenario 4). For each scenario 100 memory dumps are generated which are afterwards analyzed regarding the number of causal and VAD inconsistencies and the ability to retrieve the corresponding artifact with either a structured or unstructured analysis approach. The structured analysis methods rely on kernel data structures which are parsed during the analysis. In contrast, unstructured analysis methods scan the whole memory dump using a technique called pool tag scanning [8].

In summary, four memory acquisition tools were tested: Belkasoft's RAM Capturer[4], FTK Imager[5] and Magnet RAM Capture[6], which are closed-source but free, and WinPmem, which is open-source. As a result, each scenario shows a correlation between the execution time and the number of inconsistencies, confirming the results of Ottmann, Breitinger, and Freiling [3] and Rzepka et al. [7]. Moreover, there seems to be a relationship between the number of VAD and causal inconsistencies and how often the process can be found with structured analysis methods in scenario 1. This supports the assumption that with more inconsistencies more relevant data structures are affected which leads to a less found artifact. There were, however, no negative effects when using an unstructured analysis approach regarding the ability to find the artifact. A similar result is found in scenarios 2 and 3: the unstructured analysis method seems to be more robust against inconsistencies, whereas the structured analysis method is less effective. However, inconsistencies seem to have less influence on network connections in general, as the connection is found in 90% of the generated memory dumps.

As scenario 4 also uses an unstructured analysis approach, we would expect to either find the opened image file reliably in all memory dumps or to find the image file less frequently with a higher amount of inconsistencies. Surprisingly, neither is the case – the acquisition tool with the highest amount of inconsistencies generates memory dumps which contains the image file in almost all cases (see Table 1). In contrast, the memory dumps generated by the other three tools contain the image file in less than 50% of the dumps. In conclusion, tool 3 seems to acquire the pages in a different way compared with the other acquisition tools.

In summary, unstructured analysis approaches seem to be more robust against inconsistencies. However, unstructured analysis methods do not give any context where the information was found which can be misleading during an investigation. Moreover, the number of inconsistencies seems to influence structured analysis methods which rely on kernel data structures and are therefore more vulnerable to changes during the acquisition process.

3 Conclusion and Future Work

Section 2 summarized state of the art to measure the quality of inconsistencies typically found in Windows memory dumps. However, the proposed method does not consider inconsistencies in other kernel data structures and how these inconsistencies relate to each other. Therefore, more research in redundant information in kernel data structures is needed to find other inconsistency indicators. There is an ongoing work to investigate different paths to data structures and whether they hold information which can be used as inconsistency indicators in Linux operating systems using a kernel object graph, as proposed by Pagani and Balzarotti [5]. Moreover, the relationship between inconsistencies in other data structures and the presented VAD inconsistencies should be analyzed in order to expand on the VAD being a suitable indicator, and therefore be regarded as a reliable metric, for representing the consistency of the operating system.

The concurrency of the operating system is an important factor in generating consistent memory dumps. A straightforward method to acquire a consistent memory dump is therefore to halt the whole system and dump the memory which is called *instantaneous* consistency by Ottmann, Breitinger, and Freiling [4]. However, this is nearly impossible to achieve when the system in question is not virtualized. Hence, we suggest to research the ability of the operating system to prioritize processes and how prioritizing the memory acquisition process may influence the consistency of the generated memory dump. In particular, we currently measure the impact of adapting the process priority of the acquisition program to the quantity of inconsistencies using the open source acquisition tool WinPmem.

Additionally, research regarding the ability of memory analysis tools to extract certain artifacts is needed. We suggest to do experiments on a dataset consisting of *instantaneous* snapshots and analyze the memory dumps concerning our proposed inconsistency indicators, as well as whether other inconsistencies are present in the memory snapshots.

We point out that the ability to retrieve certain artifacts from memory, e.g. artifacts of opened image files, needs to be further researched. In particular, we need to investi-

[3] https://gitlab.com/DW0lf/fortrace
[4] https://belkasoft.com/ram-capturer
[5] https://www.exterro.com/digital-forensics-software/ftk-imager
[6] https://www.magnetforensics.com/resources/magnet-ram-capture/

	Tool 1	Tool 2	Tool 3	Tool 4	Ideal Snapshot
Mean acquisition time in min.	2.37	3.18	8.59	3.21	–
Found picture (unstructured)	49/100	49/100	99/100	45/100	10/10
No. analyzable memory dumps	92/100	56/100	78/100	90/100	10/10
No. VAD inconsistent memory dumps	92	56	78	90	0
Total VAD inconsistencies	46092	59686	119946	67146	0
Mean VAD inconsistencies	501	1066	1538	746	0
No. analyzable memory dumps	95/100	97/100	86/100	93/100	10/10
No. causally inconsistent memory dumps	71	73	74	72	0
Total causal inconsistencies	1569	1077	3628	1385	0
Mean causal inconsistencies	16	11	41	15	0

Table 1: Overview of the results of scenario 4, including in how many memory dumps the picture could be found, the number of VAD and causal inconsistencies, as well as in how many memory dumps those inconsistencies were found.[6]

gate the consequences of inconsistencies present in memory dumps and their impact on analysis tools in retrieving forensic artifacts. To do so, we need to identify the exact changes made to the pages of the relevant data structures during the memory acquisition process. We suggest to generate a series of memory snapshots, e.g. a snapshot after each x seconds, during the memory acquisition, and to compare the snapshots with each other to find any changes in the corresponding data structures.

References

[1] Andrew Case and Golden G Richard III. "Memory forensics: The path forward". In: *Digital Investigation* 20 (2017), pp. 23–33. ISSN: 1742-2876. URL: https://doi.org/10.1016/j.diin.2016.12.004.

[2] Michael Hale Ligh et al. *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. Wiley, 2014.

[3] Jenny Ottmann, Frank Breitinger, and Felix Freiling. "An Experimental Assessment of Inconsistencies in Memory Forensics". In: *ACM Trans. Priv. Secur.* 27.1 (Dec. 2023). ISSN: 2471-2566. DOI: 10.1145/3628600. URL: https://doi.org/10.1145/3628600.

[4] Jenny Ottmann, Frank Breitinger, and Felix Freiling. "Defining Atomicity (and Integrity) for Snapshots of Storage in Forensic Computing". In: *Proceedings of the Digital Forensics Research Conference Europe (DFRWS EU) 2022* (Oxford). Mar. 29–Apr. 1, 2022. URL: https://dfrws.org/presentation/defining-atomicity-and-integrity-for-snapshots-of-storage-in-forensic-computing/.

[5] Fabio Pagani and Davide Balzarotti. "Back to the Whiteboard: a Principled Approach for the Assessment and Design of Memory Forensic Techniques". In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1751–1768. ISBN: 978-1-939133-06-9. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/pagani.

[6] Lisa Rzepka et al. "A Scenario-Based Quality Assessment of Memory Acquisition Tools and its Investigative Implications". In: *Proceedings of the Digital Forensics Research Conference Europe (DFRWS EU) 2025* (Brno). Apr. 1–4, 2025. URL: tbd.

[7] Lisa Rzepka et al. "Causal Inconsistencies Are Normal in Windows Memory Dumps (Too)". In: *Digital Threats* 5.3 (Oct. 2024). DOI: 10.1145/3680293. URL: https://doi.org/10.1145/3680293.

[8] Andreas Schuster. "Pool Allocations as an Information Source in Windows Memory Forensics". In: *IT-Incident Management & IT-Forensics - IMF 2006*. Bonn: Gesellschaft für Informatik e. V., 2006, pp. 104–115. ISBN: 978-3-88579-191-1.

[9] Dennis Wolf, Thomas Göbel, and Harald Baier. "Hypervisor-based data synthesis: On its potential to tackle the curse of client-side agent remnants in forensic image generation". In: *Forensic Sci. Int. Digit. Investig.* 48 (2024), p. 301690. DOI: 10.1016/J.FSIDI.2023.301690. URL: https://doi.org/10.1016/j.fsidi.2023.301690.