The Influence of Virtual Secure Mode (VSM) on Memory Acquisition

Niken Dwi Wahyu Cahyani[1], Erwid M Jadied[2], Endro Ariyanto[4]
Informatics Faculty
Telkom University
Bandung, Indonesia

Nurul Hidayah Ab Rahman[3]
Centre for Information Security Research
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia
Parit Raja, Malaysia

***Abstract*—Recently, acquiring the Random Access Memory (RAM) full memory and access data is gaining significant interest in digital forensics. However, a security feature on the Windows operating system - Virtual Secure Mode (VSM) - presents challenges to the acquisition process by causing a system crash known as a Blue Screen of Death (BSoD). The crash is likely to occur when memory acquisition tools are being used. Subsequently, it disrupts the goal of memory acquisition since the system must be restarted, and the RAM content is no longer available. This study analyzes the implications of VSM on memory acquisition tools as well as examines to what extent its impact on the acquisition process. Two memory acquisition tools, namely FTK Imager and Belkasoft RAM Capturer, were used to conduct the acquisition process. Static and dynamic code analyses were performed by using reverse engineering techniques that are disassembler and debugger. The results were compared based on the percentage of unreadable memory between active and inactive VSM. Static analysis showed that there is no difference between all applications' functions for both active and inactive VSM. Further Bugcheck analysis of the MEMORY.DMP is pointed to the ad_driver.sys module in FTK Imager that causes the system to crash. The percentage of unreadable memory while running on active VSM and inactive VSM for Belkasoft is about 0.6% and 0.0021%, respectively. These results are significant as a reference to digital investigators as consistent with the importance of RAM dump in live forensics.**

Keywords—Live forensics; memory acquisition; virtualization; virtual secure mode

I. Introduction

As defined by the Digital Forensics Research Workshop (DFRWS), digital forensics is the use of scientifically derived and proven methods to preserve, collect, validate, analyze, and present admissible digital data that meet the court requirements [1]. Digital data originated from electronic devices that have data storage capability, including smartphones, digital cameras, and even printers. There are two types of digital data, namely: (i) volatile data – data that will be lost when there is no electrical power on the devices, and (ii) non-volatile data – data that is still stored in the device's storage media even though the power is turned off. RAM forensics or memory forensics involves collecting and examining volatile data. It becomes a priority to undertake the live acquisition if an electronic device is on, considering the data will be lost when the device is turned off. Furthermore, some cyber security incidents require RAM forensics such as malware attacks, due to its behavior that could leave no trail on non-volatile memory [2]. As an example, a study [3] was able to identify Advanced Encryption Standard (AES) keys in the memory of a ransomware process by examining memory dumps using live forensics tools. It further indicates that artifacts from memory forensics are not limited to evidence collection, yet they could be utilized to minimize the impact of cyber incidents.

While there has been significant development in advanced computing architecture, it poses challenges to memory forensics practices. For example, the use of a recent security feature known as Virtual Secure Mode (VSM), which was started from Windows 10 and Windows Server 2016 operating system, complicates the acquisition of volatile data in memory. It has been highlighted in [4] that the use of some acquisition tools (e.g., Magnet RAM Capturer, FTK Imager) to undertake live forensics causes the system to crash. Subsequently, the volatile data (i.e., the initial object of the acquisition) is no longer available since the operating system will restart the system [4], [5]. However, much work remains to be done in the technical analysis such as what happens to the system when the VSM feature is active that affects the tools during the memory acquisition process. This motivates the direction of this study to carry out further investigations on the VSM environment.

It has been noted that not all memory acquisition tools can complete the acquisition process during an active VSM environment. Therefore, this study aims to conduct a technical analysis of the VSM effects on the live memory acquisition process using two cases. The first case is a successful memory acquisition by using the Belkasoft RAM Capturer tool, and the second case is an unsuccessful memory acquisition by using the FTK Imager tool. Reverse engineering techniques are applied to analyze the behavior of the system. The main methods used are static and dynamic code analysis using IDA disassembler and Windbg debugger. Additionally, event analysis is conducted by examining event logs collected by the operating system to facilitate our understanding of the impact.

While previous works have been studying VSM and identifying the BSoD for live forensic tools, our study may become the initial research investigating the impact of VSM on live forensics. The results are discussed with technical data produced from reverse engineering techniques. Static analysis results can be used to understand the tools' program code that could directly lead to crash events. The dynamic analysis could demonstrate the tools' behavior in their running state that may (or may not) cause the impact, and how they interact with the operating system as the manager of the computer system including memory. We contribute to the underlying methodology that applies static, dynamic, and event analysis in examining the behavior of VSM and how it impacts the running memory acquisition tools.

The remaining of this paper is organized as follows: Section II discusses the related work, Section III describes the materials and methods employed in this study, Section IV discusses the results, Section V presents the conclusion, and Section VI highlights the limitations and future work.

II. Related Work

Collecting, and preserving the data of Random Access Memory (RAM) for forensic analysis is considered critical in live forensics. It contains many valuable forensic interest artifacts, including processes running on the computer. Examples of the content's use are to examine security incidents and get data from encrypted containers when it is being opened. The importance of memory forensic acquisition has attracted significant interest in recent years. Arfeen et al. [6] developed a framework for memory acquisition periodically to analyze process behavior while it is running and reside in memory to help ransomware detection. Prakoso et al. [7] examined how Metasploit attacks on Windows 10 can be analyzed using live forensics techniques on the volatile memory. The study used three well-known RAM acquisition tools, namely: FTK Imager, Dumpit, and Magnet RAM Capture. Volatility was used as the analysis tool. The results showed that RAM's live forensics can obtain key artifacts including the attacker's IP address and evidence of malware. Kazim et al. [8] identified chat artifacts of an instant messaging tool including master encryption keys that are encrypted by Bitlocker and Truecrypt, from memory dumps of Windows 7 computers. The memory dump was been analyzed using analysis tools such as Volatility and Rekall [9], [10]. The results confirm the necessity of deploying mechanisms to collect RAM from local and remote systems to support the RAM acquisition, for incident responder teams.

Choosing the appropriate tools for the acquisition and analysis of memory forensics depends upon the compatibility between digital devices and operating systems, which may pose challenges to investigators [11]. Therefore, many existing studies have attempted to understand the strengths and weaknesses of memory acquisition tools. A study in [12] compared four tools, namely Windows Memory Reader, Belkasoft's Live Ram Capturer, ProDiscover, and FTK Imager, to examine their performance in capturing memory including their ease of use. Another study in [13] showed the differences in processing time, memory usage, registry key, and DLL for FTK Imager, Belkasoft RAM Capturer, Memoryze, DumpIt, and Magnet RAM Capturer. Similarly, [14] also examined how the combination of Belkasoft RAM Capturer, FTK Imager, and Winhex can be utilized to obtain data for the Line app in Windows 8.1. Prakoso et al. [7] identified that FTK Imager, Dumpit, and Magnet RAM Capture, have the same performance in acquiring the targeted artifact of a Metasploit attack in Windows 10 based on their acquisition results comparison.

With the important role of memory acquisition and analysis in digital forensics, it indicates that any issues that may hinder these processes shall be examined, including VSM. VSM is a Windows 10 technology for creating and managing a secure operating system environment [15]. The secure environment is designed to be a place for the execution of critical security functions, protecting it from attacks directed against the operating system. VSM uses virtualization as its base [16].Virtualization on a machine run by an emulator, commonly known as a hypervisor. Microsoft gives a particular name to its hypervisor system which is Hyper-V, while the virtual machine is known as a partition (e.g., Partition A and Partition B). Hyper-V virtualizes hardware resources for each partition and manages these virtual resources, including virtual memory and CPU.

Details architecture of a Windows environment that supports VSM shows that Hyper-V occupies the root partition [16]. The partition houses two environment modes, namely: (i) kernel, and (ii) user. Each environment operates on a separate domain, called the Virtual Trust Level (VTL). VTL enforces isolation in three aspects. First is memory access in which each VTL has a set of memory access protections that prevent an allocated VTL's memory from being accessed by entities in another VTL. Second is the virtual processor state, where each VTL has a set of private virtual processor registers associated with it. Third, interrupt in which each VTL has a separate interrupt system to prevent interference from entities operating in other VTLs during sending and processing of interrupts.

A study on Windows 10 reported that Hyper-V implements two VTLs: VTL 0 and VTL 1[16]. VTL 0 hosts a traditional Windows environment. Users running in VTL 0 are referred to as normal environment users, while the running kernels are known as normal kernels. VTL 1, on the other hand, is the place for the Windows environment to perform security-critical functionality. The environment is referred to as a safe environment. VTL-based memory access protection enforced by Hyper-V can be further referred to in [16]. The study shows the memory region's contents that are part of the memory dump of a VSM-enabled Windows environment mapped to the lsaIso.exe trustlet. The question mark character ('?') indicates unreadable memory because it cannot be accessed beyond the isolation limits implemented by VTL 1, where lsaIso.exe operates. A report in [4] discussed the effect of VSM that causes BSoD on several tools including Magnet RAM Capture v1.1.1 and FTK Imager Lite v3.1.1, however other tools such as Belkasoft RAM Capturer and Passmark osforensics v 5.1.1001 were not included.

There are existing studies that have demonstrated static code analysis and dynamic code analysis. For example, a study by Hirst [17] showed an acquisition test on no-quiescent virtual machines that utilized dynamic code analysis. Another study in [18] identified memory acquisition challenges that misuse two architectural features, which are physical address layout and secure container. The authors acknowledged them as a new class of anti-memory forensic techniques. Significantly, these studies provided key guidance on the methods that can be referred to conduct testing and observation.

Yehuda et al. [19] proposed a hypervisor-based memory acquisition tool by extending the Volatility framework and implementing it in ARM64-bit kernels. The authors showed how their proposed tool can reduce the processor's consumption, maintain the coherent state of the memory dump, and generate fewer tradeoffs for network and disk acquisition. The tool successfully conducts memory acquisition without facing any difficulties caused by security and privilege levels in Linux OS and ARM processors, called Trust Zone which divides accesses into secure and non-secure ones.

Nevertheless, the study on technical analysis of the VSM effect on the live memory acquisition process is still limited. There have been significant studies of VSM architecture on Windows 10, including the details of VSM initialization activity performed by the Windows loader during the boot process, and the communication interface on VSM [16], [20]. However, the explanation still lacks the technical impacts of VSM on the memory acquisition process. In this study, therefore, we attempt to examine the memory acquisition in Windows-based OS, especially in Windows 10 that enabled the VSM feature in Intel machines to manage virtual trust levels for kernel and user processes.

III. RESEARCH METHOD

This study applied reverse engineering as it is a widely recognized technique in digital forensics to process and interpret data [21], [22]. The methods used to analyze the impact of VSM on the memory acquisition tool are static and dynamic code analysis using the IDA disassembler and windbg debugger tools.

Event analysis is conducted using the operating system's event logs for further correlation with the findings from the static and dynamic code analysis. The complete research stages are presented in Fig. 1. The hardware and software specifications used in this research are presented in Tables I and II, respectively.

Experiments in this study are conducted in two environments, (1) VSM-enabled, and (2) non-VSM-enabled. VSM feature is enabled through the BIOS by setting the "Intel Virtualization Technology" option to "Enabled." The BIOS used in this study is from the American Megatrends vendor, version 309, with VBIOS Version 1054.I021x441UAR.002.

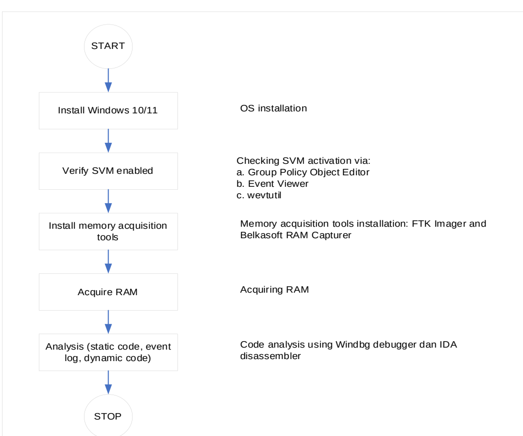

Fig. 1. Research Stages.

TABLE I. HARDWARE SPECIFICATION

No	Name	Specification	Function
1	Processor	Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz 2.30 GHz	To execute a program
2	RAM	20,0 GB (19,9 GB usable)	To store data and instructions for a process

TABLE II. SOFTWARE SPECIFICATION

No	Name	Specification	Function
1	Microsoft Windows 10	Edition: Windows 10 Enterprise Version: 21H1 OS build: 19043.1526 Experience: Windows Feature Experience Pack 120.2212.4170.0	Operating System
2	AccessData FTK Imager	Version 4.5.0.3	Acquisition Tools
3	Belkasoft RAM Capturer	Modified date 22/10/2018	Acquisition Tools
4	IDA PRO	7.5 SP3 x64	Disassembler
5	Diaphora	Version 2	Program diffing tool
6	Windbg Preview	Version 1.2202.7001.0	Debugger
7	Event Viewer	Version 1.0	Event Viewer

The steps taken to obtain data to be analyzed are presented in Table III.

TABLE III. EXPERIMENTS AND ANALYSIS STEPS

Static Code Analysis		
No		***Steps***
1		VSM feature setting [Active/Non-Active]
	a	Disassembling executable files of memory acquisition apps (.exe) [FTK Imager/Belkasoft RAM Capturer] by using IDA PRO
	b	Running the diffting plugin to compare the apps' functions in VSM vs. non-VSM environments by using Diaphora on IDA PRO
2		Save the results
Note: 4 (four) files were produced: Assembly codes for: • FTK Imager • Belkasoft RAM Capturer SQLite files for: • FTK Imager • Belkasoft RAM Capturer		
Event Log Analysis		
No		Steps
1		Acquiring memory by using FTK Imager and Belkasoft RAM Capturer in active VSM and non-active VSM environments
2		Opening Event Viewer
3		Copying event log:
	a	Event Application
	b	Event Security
	c	Event System
Note: 3 (three) files were produced: Application, Security dan System event logs		
Dynamic Code Analysis		
No		Steps
1		Choosing the target executable files
2		Preparing the required symbols
3		Starting the debugging

IV. RESULTS AND DISCUSSION

This section presents the results of experiments and discusses the memory acquisition, static code, event log, and dynamic code analyses that have been carried out.

A. Memory Acquisition Analysis

It has been observed that the FTK Imager has successfully acquired a memory dump in a non-VSM-enabled. On the other hand, no memory dump was generated when VSM was active because the system experienced a Blue Screen of Death (BSoD). Meanwhile, Belkasoft RAM Capturer managed to acquire memory dump in both VSM environments. Therefore, this section will analyze the differences in the results of memory acquisition from the FTK Imager application in a non-enabled- VSM environment (non-VSM), and Belkasoft RAM Capturer in both VSM environments.

All three memory dumps generated by the memory acquisition applications have the same size according to the measured memory capacity of 21.4 GB (23,068,672,000 bytes). Here, we will focus on the contents of the memory dump, which has the value "?????????????????...??????" as a mark of memory locations that are not readable by applications (see Fig. 2).

The data shows that the unreadable memory space of active VSM is larger than non-VSM (the sign … refers to the other 65 rows that are not displayed). It indicates that VSM enforces more limitations on physical memory access than non-VSM. The limitation can be correlated with the implementation of memory access protection for each VTL, especially for VTL1 which runs in a safe environment [16].

By calculating the portion of memory with the value "?????????????????...??????" (see Fig. 2), there are 122 MB of memory size for Belkasoft in VSM-enabled mode. It is about 0.6% of the memory size. Meanwhile, Belkasoft and FTK Imager in non-VSM mode are 0.4 MB and 0.5 MB, respectively. It is only about 0.0021% and 0.0025% of the memory size. The comparison of the unreadable memory percentage between VSM-enabled and non-VSM-enabled is significant, as consistent with the importance of memory data for a live forensic investigation.

B. Static Code Analysis

The analysis compares the differences between an active VSM state and when VSM is not active. In this section, the experiment results are grouped based on two types of memory acquisition applications tested, namely FTK Imager and Belkasoft RAM Capturer. The assembly codes are derived from the machine code from the disassembler process using IDA PRO.

A python IDA plugin called Diaphora is used to generate an SQLite file that lists the functions identified from the assembly code generated by IDA PRO. The main purpose of using Diaphora is to examine differences in the functions of the FTK Imager application for active and inactive VSM conditions and the Belkasoft RAM Capturer application.

(a) FTK Imager

NON-VSM
0003BC410-0003C5FFF
140B88220-140B8831F
1463D8220-1463D8342
148A02000-148A03FFF
148AFB410-148AFBFFF
14A5FC000-14A5FCFFF
14A5FE000-14A5FFFFF
14A601000-14A601FFF
18A319000-18A319FFF
...
540DD5000-540DD9FFF
540DDB000-540DDC649
541DCB000-541DCBFFF
5459C6000-5459CC120
5459CC181-5459CFFFF
5479D3000-5479D3FFF
5479DA000-5479DAFFF

(b) Belkasoft RAM Capturer

VSM	NON-VSM
0002C2000-0002C5FFF	0003BC410-0003C5FFF
0003BD000-0003CFFFF	0009D0000-0009D0FFF
1065AC000-1065ACFFF	002CDB000-002CDBFFF
1173D0000-1173D0FFF	002DD6000-002DD6FFF
1187D9000-1187D9FFF	0050CA000-0050CAFFF
1189C6000-1189C7FFF	0051C9000-0051CCFFF
118BC8000-118BC9FFF	0052CD000-0052CDFFF
118BD1000-118BD1FFF	0056C6000-0056C6FFF
118FCA000-11BFCAFFF	0057C8000-0057C8FFF
...	...
44D82B000-44D82BFFF	545DD3000-545DD3FFF
44DC28000-44DC28FFF	54CC41EA0-54CC41F9F
44DC2C000-44DC2CFFF	
450D4B410-450D4BFFF	
451267000-451271649	
463EB4000-463EBBFFF	
4AF92D2E0-4AF92DFFF	
4AFF55000-4AFF55FFF	
4B0C50000-4B0C50FFF	
4B1C763D0-4B1C76FFF	
4B5198000-4B5198FFF	
4B7087610-4B7087FFF	
4B718C4D0-4B718DFFF	
4B72993A0-4B7299FFF	
4BB415000-4BB415FFF	
4FA01D000-4FA01DFFF	
50CFA8610-50CFA8FFF	
537CBC410-537CCDFFF	
53C4AA320-53C5C4FFF	
559E16EA0-559E16F9F	

Fig. 2. Memory Sectors that are Unreadable by FTK Imager and Belkasoft RAM Capturer.

Diaphora succeeded in recognizing 32483 functions from the assembly code of the FTK Imager application. The results were compared in terms of names, order of contents, hash values, and relative virtual address (RVA) values of all these functions. It is identified that all functions of these assembly codes are equal when run in both enabled VSM and non-enabled VSM. The number of files with the status of "100% equal", "Perfect match, same name," "Same order and hash," and "Same RVA and hash" are 21760, 3, 6507, and 4213 files, respectively. Likewise, for the 38 functions that have been recognized from the Belkasoft RAM Capturer application, all of them are also identified to be the same. It has been observed that 15 files with the status of "100% equal", 3 files with the status of "Perfect match, same name," six files with the status of "Same order and hash," and 14 files with the status of "Same RVA and hash".

Based on these findings, it can be deduced that FTK Imager and Belkasoft RAM Capturer applications do not have different functions in their static code for both environments (i.e., VSM is enabled and non-enabled). This further indicates that the VSM environment does not affect the overall running characteristics of the application.

C. Event Log Analysis

Windows operating system generates three event logs which are Application, System, and Security logs. In this study, we focused on observing events from Application and System logs because they contain key information when the system crashes and restarts.

We identified the records of events associated with FTK Imager application crashes during the memory acquisition process, and when VSM is enabled from the event Application log. Detailed information is presented in Fig. 3. It describes the error name called BlueScreen and informs that this crash event has the data stored in the MEMORY.DMP file.

We observed more information from the System log events. Fig. 4 reports an event with an "error" status. This status is captured from the second experiment scenario; when the VSM is not activated. Detailed information can be found in the General field, stating that the VSM feature is not activated and the Hypervisor as a virtualization emulator fails to run. This information confirms the environment in which we did not activate the VSM. While this setting can be checked from the BIOS configuration, this "error" status notified us that this virtualization-based enablement policy should be mandatory in Windows 10. This situation may lead to anti-forensics, where the implementation of security control prevents digital forensic tools to operate.

The captured information about the error when the FTK Imager is running on the active VSM is presented in Fig. 5.

It is likely indicating the cause of the blue screen and the record of the crash event that forced the system to reboot. The operating system provides the information in their Bugcheck error in Event Viewer. Bugcheck error will record the BSoD event, and its basic error code to identify what caused the BSoD. The fourth row in this Bugcheck provides information that the system is rebooted and IsolatedUserMode is active.

Level	Date&Time	Source	Event ID	Task Category
Information	14/01/2022 11:56	Windows Error Reporting	1001	None
General				

```
Fault bucket , type 0
Event Name: BlueScreen
Response: Not available
Cab Id: 0
Problem signature:
P1: 3b
P2: c0000005
P3: fffff8054f3339d0
P4: ffffc2016e476c10
P5: 0
P6: 10_0_19043
P7: 0_0
P8: 256_1
P9:
P10:
Attached files:
\\?\C:\Windows\Minidump\011422-7343-01.dmp
\\?\C:\Windows\TEMP\WER-16078-0.sysdata.xml
\\?\C:\Windows\MEMORY.DMP
\\?\C:\ProgramData\Microsoft\Windows\WER\Temp\WER759D.tmp.WERInternalMetadata.xml
\\?\C:\ProgramData\Microsoft\Windows\WER\Temp\WER75CD.tmp.xml
\\?\C:\ProgramData\Microsoft\Windows\WER\Temp\WER761A.tmp.csv
\\?\C:\ProgramData\Microsoft\Windows\WER\Temp\WER7679.tmp.txt
These files may be available here:
\\?\C:\ProgramData\Microsoft\Windows\WER\ReportQueue\Kernel_3b_8ee97460173b5dfa1496a56646ae5f378cb4a4d6_00000000_6
4ca668f-ffeb-43c1-a4da-33e8c9aec794
Analysis symbol:
Rechecking for solution: 0
Report Id: 7e2accb6-516b-4ad0-af85-c3c80b9a909e
Report Status: 4
Hashed bucket:
Cab Guid: 0
```

Fig. 3. Selected Significant Events of FTK Imager in Non-Active VSM.

	Level	Date&Time	Source	Event ID	Task Category
1	Level	Date&Time	Source	Event ID	Task Category
	Error	14/01/2022 12:34:55	Kernel-Boot	124	-80
	General				
	The virtualization-based security enablement policy check at phase 0 failed with status: Virtual Secure Mode (VSM) is not initialized. The hypervisor or VSM may not be present or enabled.				
2	Level	Date&Time	Source	Event ID	Task Category
	Error	14/01/2022 12:34:55	Hyper-V-Hypervisor	41	None
	General				
	Hypervisor launch failed; Either VMX not present or not enabled in BIOS.				

Fig. 4. Selected Significant Events of FTK Imager in Non-Active VSM.

	Level	Date&Time	Source	Event ID	Task Category
1	Level	Date&Time	Source	Event ID	Task Category
	Error	14/01/2022 11:56:37	BugCheck	1001	None
	General				
	The computer has rebooted from a bugcheck. The bugcheck was: 0x0000003b (0x00000000c0000005, 0xfffff8054f3339d0, 0xffffc2016e476c10, 0x0000000000000000). A dump was saved in: C:\Windows\MEMORY.DMP. Report Id: 7e2accb6-516b-4ad0-af85-c3c80b9a909e.				
2	Level	Date&Time	Source	Event ID	Task Category
	Error	14/01/2022 11:56:25	TPM	15	None
	General				
	The device driver for the Trusted Platform Module (TPM) encountered a non-recoverable error in the TPM hardware, which prevents TPM services (such as data encryption) from being used. For further help, please contact the computer manufacturer.				
3	Level	Date&Time	Source	Event ID	Task Category
	Critical	14/01/2022 11:56:24	Kernel-Power	41	-63
	General				
	The system has rebooted without cleanly shutting down first. This error could be caused if the system stopped responding, crashed, or lost power unexpectedly.				
4	Level	Date&Time	Source	Event ID	Task Category
	Information	14/01/2022 11:56:22	IsolatedUserMode	3	None
	General				
	Secure Kernel started with status STATUS_SUCCESS and flags 0.				
5	Level	Date&Time	Source	Event ID	Task Category
	Warning	14/01/2022 11:56:22	Hyper-V-Hypervisor	157	None
	General				
	The hypervisor did not enable mitigations for CVE-2018-3646 for virtual machines because HyperThreading is enabled and the hypervisor core scheduler is not enabled. To enable mitigations for CVE-2018-3646 for virtual machines, enable the core scheduler by running "bcdedit /set hypervisorschedulertype core" from an elevated command prompt and reboot. "CVE-2018-3646: Systems with microprocessors utilizing speculative execution and address translations may allow unauthorized disclosure of information residing in the L1 data cache to an attacker with local user access with guest OS privilege via a terminal page fault and a side-channel analysis."				
6	Level	Date&Time	Source	Event ID	Task Category
	Information	14/01/2022 11:56	Hyper-V-Hypervisor	165	None
	General				
	Hypervisor configured mitigations for CVE-2019-11091, CVE-2018-12126, CVE-2018-12127, CVE-2018-12130 for virtual machines. Processor not affected: false Processor family not affected: false Processor supports microarchitectural buffer flush: true Buffer flush needed: true				

Fig. 5. Selected Significant Events of FTK Imager in Active VSM.

We have the same observation about the active IsolatedUserMode when Belkasoft RAM Capturer runs in active VSM (see Fig. 6). Other key points in Fig. 5 are shown in rows 5 and 6. These two rows indicate a Hypervisor failure to handle CVE-2018-3646. Further examination of Common Vulnerabilities and Exposure (CVE) suggests that the vulnerability is related to the possibility of unauthorized disclosure of information [23]. A possible explanation for this failure could be associated with the existence of a memory space isolation system that caused the memory acquisition tools unable to access the information.

An additional analysis of the MEMORY.DMP file was undertaken to obtain further information on the "Bugcheck" event. We used the Windbg application and ran the command !analyze -v (see Fig. 7). The Bugcheck analysis was carried out on the MEMORY.DMP file supports that the crash is related to the FTK Imager application. The associated module is ad_driver, and the image name is ad_driver.sys. The file directory is located at C:\Users\[UserName]\AppData\Local\Temp. This is consistent with the information on the BSoD screen, which indicates an error has occurred in the driver.sys. Furthermore, Windbg provides more information about this error by indicating that the driver.sys in question is related to ad_driver.

D. Dynamic Code Analysis

Dynamic code analysis examines the application's behavior while the operating system executes it. Interaction from the user will affect the direction of execution. The dynamic code analysis is performed on the FTK Imager application with an active VSM environment. The aim is to observe the application's behavior related to the BSoD error.

	Level	Date&Time	Source	Event ID	Task Category
1	Information	14/01/2022 11:58:35	IsolatedUserMode	5	None
	General				
	Secure Trustlet NULL Id 0 and Pid 0 started with status STATUS_SUCCESS.				

Fig. 6. Selected Significant Events of Belkasoft RAM Capturer in Active VSM.

```
*******************************************************************************
*                        Bugcheck Analysis                                    *
*******************************************************************************
FILE_IN_CAB:  MEMORY.DMP
BUGCHECK_CODE:  3b
BUGCHECK_P1: c0000005
BUGCHECK_P2: fffff803362939d0
BUGCHECK_P3: ffffab8f63b96c10
BUGCHECK_P4: 0
CONTEXT:  ffffab8f63b96c10 -- (.cxr 0xffffab8f63b96c10)
rax=0000000000000000 rbx=0000000000000000 rcx=ffffd302a40fe000
rdx=00002cfd5c603000 rsi=0000000002600000 rdi=0000000000700000
rip=fffff803362939d0 rsp=ffffab8f63b97618 rbp=ffffd302a40fd000
 r8=0000000000001000  r9=0000000000000080 r10=7ffffffffffffffc
r11=ffffd302a40fd000 r12=ffffffffffffffff r13=0000000000001000
r14=ffffd302b2dbc160 r15=ffffd302b4030a10
iopl=0         nv up ei ng nz na pe cy
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00050283
ad_driver+0x39d0:
fffff803`362939d0 488b440af8      mov     rax,qword ptr [rdx+rcx-8] ds:002b:00000000`00700ff8=????????????????
Resetting default scope

PROCESS_NAME:  FTK Imager.exe
STACK_TEXT:
ffffab8f`63b97618 fffff803`3629256a     : 00000000`c00000bb fffff803`3a8e681c ffffab8f`63b976a0 01d8134d`cdf32b29 : ad_driver+0x39d0
ffffab8f`63b97620 fffff803`36291110     : 00000000`02600000 fffff803`34ff5f01 00000000`00700000 00000000`000008c4 : ad_driver+0x256a
ffffab8f`63b976b0 fffff803`34c8f825     : ffffd302`b31fe0a0 ffffd302`00000000 00000000`00000002 00000000`00000001 : ad_driver+0x1110
ffffab8f`63b97700 fffff803`35075b58     : ffffab8f`63b97a80 ffffd302`b31fe0a0 00000000`00000001 ffffd302`b40020c0 : nt!IofCallDriver+0x55
ffffab8f`63b97740 fffff803`35075957     : 00000000`00000000 ffffab8f`63b97a80 00000000`00000000 ffffab8f`63b97a80 : nt!IopSynchronousServiceTail+0x1a8
ffffab8f`63b977e0 fffff803`35074cd6     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!IopXxxControlFile+0xc67
ffffab8f`63b97920 fffff803`34e08cb5     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!NtDeviceIoControlFile+0x56
ffffab8f`63b97990 00007ffa`9ef0ce54     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiSystemServiceCopyEnd+0x25
00000000`006f6918 00000000`00000000     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : 0x00007ffa`9ef0ce54
SYMBOL_NAME:  ad_driver+39d0
MODULE_NAME: ad_driver
IMAGE_NAME:  ad_driver.sys
STACK_COMMAND:  .cxr 0xffffab8f63b96c10 ; kb
BUCKET_ID_FUNC_OFFSET:  39d0
FAILURE_BUCKET_ID:  AV_ad_driver!unknown_function
OS_VERSION:  10.0.19041.1
BUILDLAB_STR:  vb_release
OSPLATFORM_TYPE:  x64
OSNAME:  Windows 10
FAILURE_ID_HASH:  {8f6b899e-895f-35a5-567c-c877346fcd6e}
Followup:     MachineOwner
---------
```

Fig. 7. Summary of Bugcheck Analysis Results.

The analysis commenced by selecting the "Start debugging" menu in the Windbg Preview application and selecting the executable file from the FTK Imager application. The debugger downloaded the symbol file "ProfUISad64.pdb" to perform the debugging process. The following commands are typed on the "Command" page to control the process:

- To load symbols:
  - .symfix
  - .reload
- To run the FTK Imager application:
  - g

As a result of executing those commands, we identified that the last module before the system crash was C:\Windows\system32\mssprxy.dll. The module is recorded from the debugger as a module that is loaded before the user clicks the "Capture Memory" button. This is an unexpected finding because the information from the event log analysis suggests the module that caused the crash is ad_driver.sys. Therefore, other scenarios in dynamic code analysis shall be considered to find the very last module loaded by the operating system before the crash happens.

V. Conclusion

This study aims to conduct a technical analysis of the effects of VSM on the memory acquisition process. Two cases were observed that are: (1) a successful acquisition process by using the Belkasoft RAM Capturer, and (2) an unsuccessful acquisition process by using the FTK Imager. The static analysis results of the two applications did not show any differences in the program code when the application machine code disassembler was carried out, both when VSM was enabled and non-enabled. It is concluded that the VSM environment does not affect the program modules of the application.

Meanwhile, Application event analysis comprises logs of system crashes and is stored in the MEMORY.DMP file. Bugcheck analysis of the dump file shows the cause of the system experiencing BSoD when it executes the ad_driver.sys module. Furthermore, results from dynamic analysis explained the behavior of the FTK Imager application just before the BSoD occurs, and it is identified that the application accesses the C:\Windows\system32\mssprxy.dll module.

VI. Future Work

This study highlights the impact of VSM on the memory acquisition process that causes the loss of memory artifacts when the process is halted and the system restarts. However, this study is limited to two memory acquisition tools running on the Windows operating system, which respond differently to the activation of the VSM feature. More importantly, the difference opens more directions for future work. Investigating the impact on other tools and operating systems would present more significant results to be compared. Testing environments should involve different scenarios in dynamic code analysis and conduct an in-depth analysis of the ad_driver.sys module content. This is to seek further understanding of how the module causes the system crashes.

Acknowledgment

This research was supported by the Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2020/ICT07/UTHM/03/1). The authors would like to thank Telkom University and Universiti Tun Hussein Onn Malaysia for their assistance, and anonymous reviewers for their constructive and generous feedback.

References

[1] Collective work of all attendees, “Digital Forensic Research Workshop,” in Proceedings of The Digital Forensic Research Conference (DFRWS) (2001), Aug. 2001.

[2] A. Case and G. G. Richard, “Memory forensics: The path forward,” Digit Investig, vol. 20, pp. 23–33, 2017, doi: https://doi.org/10.1016/j.diin.2016.12.004.

[3] S. R. Davies, R. Macfarlane, and W. J. Buchanan, “Evaluation of live forensic techniques in ransomware attack mitigation,” Forensic Science International: Digital Investigation, vol. 33, p. 300979, 2020, doi: https://doi.org/10.1016/j.fsidi.2020.300979.

[4] J. Hale, “Memory Acquisition and Virtual Secure Mode,” https://df-stream.com/2017/08/memory-acquisition-and-virtual-secure/, 2017.

[5] H. K. Brendmo, “Live forensics on the Windows 10 secure kernel,” Norwegian University of Science and Technology, 2017.

[6] A. Arfeen, M. Asim Khan, O. Zafar, and U. Ahsan, “Process based volatile memory forensics for ransomware detection,” Concurr Comput, vol. 34, no. 4, Feb. 2022, doi: 10.1002/cpe.6672.

[7] D. C. Prakoso, I. Riadi, and Y. Prayudi, “Detection of Metasploit Attacks Using RAM Forensic on Proprietary Operating Systems,” Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, pp. 155–160, May 2020, doi: 10.22219/kinetik.v5i2.1037.

[8] A. Kazim, F. Almaeeni, S. al Ali, F. Iqbal, and K. Al-Hussaeni, “Memory Forensics: Recovering Chat Messages and Encryption Master Key,” in 2019 10th International Conference on Information and Communication Systems (ICICS), Jun. 2019, pp. 58–64. doi: 10.1109/IACS.2019.8809179.

[9] S. Anson, Ed., “Acquiring Memory,” in Applied Incident Response, Wiley, 2019, pp. 103–131. doi: 10.1002/9781119560302.ch5.

[10] S. Anson, Ed., “Memory Analysis,” in Applied Incident Response, Wiley, 2019, pp. 235–275. doi: 10.1002/9781119560302.ch9.

[11] G. M. Jones and S. G. Winster, “An Insight into Digital Forensics: History, Frameworks, Types and Tools,” in Cyber Security and Digital Forensics, Wiley, 2022, pp. 105–125. doi: 10.1002/9781119795667.ch6.

[12] R. J. McDown, C. Varol, L. Carvajal, and L. Chen, “In-Depth Analysis of Computer Memory Acquisition Software for Forensic Purposes,” J Forensic Sci, vol. 61, pp. S110–S116, Jan. 2016, doi: 10.1111/1556-4029.12979.

[13] M. N. Faiz and W. A. Prabowo, “Comparison of Acquisition Software for Digital Forensics Purposes,” Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, pp. 37–44, Nov. 2018, doi: 10.22219/kinetik.v4i1.687.

[14] I. Riadi, S. Sunardi, and M. E. Rauli, “Live Forensics Analysis of Line App on Proprietary Operating System,” Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, pp. 305–314, Oct. 2019, doi: 10.22219/kinetik.v4i4.850.

[15] Microsoft, “Virtual Secure Mode,” Microsoft Learn, Jul. 07, 2022. https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/tlfs/vsm (accessed Oct. 05, 2022).

[16] A. Milenkoski and D. Phillips, “Virtual Secure Mode: Architecture Overview,” 2019.

[17] N. W. Hirst, “The implications of virtual machine introspection for digital forensics on nonquiescent virtual machines,” NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2011.

[18] N. Zhang, R. Zhang, K. Sun, W. Lou, Y. T. Hou, and S. Jajodia, “Memory Forensic Challenges under Misused Architectural Features,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 9, pp. 2345–2358, Sep. 2018, doi: 10.1109/TIFS.2018.2819119.

[19] R. ben Yehuda, E. Shlingbaum, Y. Gershfeld, S. Tayouri, and N. J. Zaidenberg, “Hypervisor memory acquisition for ARM,” Forensic Science International: Digital Investigation, vol. 37, Jun. 2021, doi: 10.1016/j.fsidi.2020.301106.

[20] A. Milenkoski, “Virtual Secure Mode: Communication Interfaces,” 2019.

[21] A. M. Marshall and R. Paige, “Requirements in digital forensics method definition: Observations from a UK study,” Digit Investig, vol. 27, pp. 23–29, 2018, doi: https://doi.org/10.1016/j.diin.2018.09.004.

[22] R. Nordvik, H. Georges, F. Toolan, and S. Axelsson, “Reverse engineering of ReFS,” Digit Investig, vol. 30, pp. 127–147, 2019, doi: https://doi.org/10.1016/j.diin.2019.07.004.

[23] National Vulnerability Database, “CVE-2018-3646,” The MITRE Corporation, Dec. 28, 2017. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-3646 (accessed Oct. 05, 2022).