

ANALYSIS OF THE EFFECT OF VSM ON THE MEMORY ACQUISITION PROCESS USING THE DYNAMIC ANALYSIS METHOD

Sinta Nur Maulina*¹⁾, Niken Dwi Wahyu Cahyani²⁾, Erwid Musthofa Jadied³⁾

1. Telkom University, Indonesia
2. Telkom University, Indonesia
3. Telkom University, Indonesia

Article Info

Kata Kunci : Live Forensics, VSM, Tools, Crash

Keywords : Live Forensics, VSM, Tools, Crash

Article history:

Received 22 December 2022

Revised 29 December 2022

Accepted 4 January 2023

Available online 1 June 2023

DOI :

<https://doi.org/10.29100/jupi.v8i2.3745>

* Corresponding author.

Corresponding Author

E-mail address:

sintanurmaulina@student.telkomuniversity.ac.id¹

ABSTRACT

At first, forensics was restricted to studying data that was stored on a system's hard disk. However, as storage capacity and data encryption increased, applying conventional digital forensic procedures became more challenging. As a result, memory forensics techniques are developed, or are frequently referred to as live forensics, because the process is quicker and more sophisticated. Volatile memory forensics, often known as live forensics, are necessary for this condition. Live forensics has flaws, specifically that some programs can fail when the computer is in active VSM (virtual secure mode). This results in the retrievable evidence being lost. Therefore, determining the cause is essential. The software-based memory acquisition tools Autopsy, Isobuster, DumpIt, and Magnet RAM Capturer are just a few examples. According to the findings of the experiments, the tools that have crashed include DumpIt v1.3.2.20110401. A dynamic code analysis using WindBg as a tool was utilized to study the impact of VSM on the memory acquisition tool. This study's goal is to identify the instances of crashes in various forensic instruments, which will be highly useful for forensic experts performing investigations.

I. INTRODUCTION

Digital forensics or forensics is used to examine digital evidence when handling a case that requires the handling and identification of digital goods in forensic science, especially to investigate the discovery of digital device content, and is often associated with crime [1]. Digital investigators use information on an attacker's computer to find clues that can help in proving a case. One aspect is digital evidence that can be retrieved from main memory (RAM), which includes immediate information about the currently running program.

Computer forensics is an investigation and computer analysis technique that involves the stages of identification, preparation, extraction, documentation and interpretation of the origin of the data on the computer to serve as evidence of cybercrime incidents [2]. There is a problem in live forensics, namely some tools crash when the computer is in active VSM (virtual secure mode) using a 64-bit operating system, x64-based processor. This causes the evidence to be taken to be lost. Therefore, it is necessary to find the cause. There are several software-based memory acquisition tools, namely autopsy, isobuster, DumpIt, Magnet RAM Capturer. From the results of the experiments that have been carried out, the tools that have crashed are DumpIt v1.3.2.20110401.

VSM is a Hyper-V container that isolates the lsass.exe process from a running Windows 10 machine. Reduces the risk of credentials from a computer using a tool namely mimikatz, and is used for pass-the-hash attacks. Something worth mentioning is that VSM only protects domain[3] credentials. Each partition contains an operating system environment. If windows based, this environment has this architecture consists of the following parts of Windows i.e. system support processes, services, applications, Windows subsystem, Hardware abstraction layer kernel drivers. Each partition works with its own isolation abuts. The separator boundaries between partitions are created and managed by the hypervisor. Isolation partitioning is implemented so that the hypervisor allocates a separate virtual memory space. The hardware resources for each of these partitions mean that the partition is not accessible to the memory that another partition allocates. In a virtualized environment based on Hyper-V, it is managed using a partition called the root partition. Serves other partitions co-located with it. For example, the root partition hosts virtualization services. Provided by the hypervisor to make this service available on other shared partitions. Also this root partition can host device drivers because it is the only partition that has direct access to

hardware resources[4].

Hypercalls implements the services that the hypervisor displays to partitions. It involves critical system services enabling the operation of virtual systems, namely memory management services. Each Hyper-V hypercall can be uniquely identified by an identification number, which is referred to as a dialing code. An important prerequisite for a Hyper-V hypercall to be called is the existence of the hypercall page in the context of the partition. The hypercall page is a memory page that stores code to invoke a hypercall according to the Hyper-V specification. This page is exposed by the hypervisor to every partition. The Windows hypervisor is the bridge through which Hyper-V communicates with the hardware. Of course, the hardware is designed and certified to run on the Windows Server operating system. Hyper-V manages virtual machines with hardware partitions. Called a virtual partition. A virtual partition consists of a parent partition and child partitions. Partition for where Windows Server resides. Meanwhile, sub-partitions can be shared with other operating systems can be seen in figure 1.

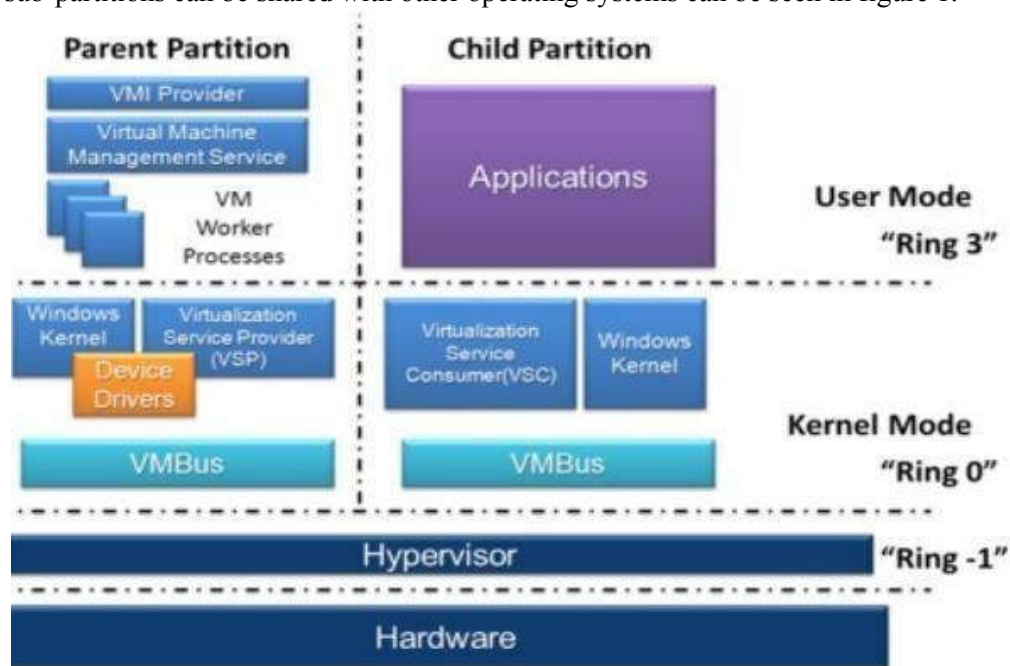


Figure 1 : Hyper-v architecture

WinDbg is a multipurpose debugger for the Microsoft Windows computer operating system, distributed by Microsoft. Debugging is the process of finding and resolving errors in a system in computing that also includes exploring the internal operations of software as an aid to development. It can be used to debug user mode applications, device drivers, and the operating system itself in kernel mode. This type of archive includes a minimum amount of information. It contains only the BSOD error message, information about the driver, the processes that were active at the time of the crash, and which kernel process or thread caused the crash in the generally small kernel memory dump, 1/3 the amount of physical memory. Kernel memory dumps are more specific than minidumps. It contains kernel mode drivers and programs, including memory allocated to the Windows kernel and hardware abstraction layers, and memory allocated to other kernel mode drivers and events. Complete memory dump. largest size and requires memory equivalent to your system RAM plus the 1 MB required by Windows to build this file. Automatic memory dump. sync with kernel memory dumps in case of issues. These differ only in how much space is used to form the dump file. This archive type does not exist in Windows 7. It was added in Windows 8. The memory disposal area is active. This type of filter element cannot determine the cause of the system failure. Windbg is the most powerful debugging and reverse engineering tool on the Windows platform. Windbg, namely X-ray plus MRI plus CT scan of programs running on the Windows operating system, including the operating system itself. It finds the cause of complex problems with programs running in Windows (OS) and programs running in Windows (OS)[5].

Dumplt is a collection of two tools, namely win32dd and win64dd, combined as an executable used to acquire physical memory. Dumplt is designed to be administered to non-technical users using a removable USB drive. Dumplt will take a snapshot from physical memory and save it to the folder[6]. Memory Dump is carried out for the purpose of memory acquisition which has two approaches for performing memory acquisition, namely hardware-based and software-based. There are so many software available to get memory privately. This software software can capture RAM privately. DumpIt is a compact portable software that makes it easy to store the contents

of physical memory[7].

Memory acquisition is the process of acquiring volatile memory (RAM) to non-volatile storage (files on disk)[8]. live forensics is a way in a forensic process where the system is still running, this is done because if the system dies then there will be lost data or information[9]. The live forensic method is usually used for cases where there is volatile data where the data will be lost if the power source dies, volatile data is usually stored in temporary media, namely RAM. Meanwhile, live forensics is used to collect data when the affected system is still alive[10]. Virtual forensic investigations mainly rely on data stored on storage media along with primary storage. Volatile memory or random access memory can store information i.e. running processes, incognito browsing sessions, clipboard statistics, information stored in plain text reports.

This study aims to analyze the effect of VSM on the live memory acquisition process using the dynamic method using the windBbg method. The Windows Debugger (WinDbg) can be used to debug kernel mode and user mode code, analyze crash dumps, and inspect CPU registers while code is running[11].

II. RESEARCH METHOD

A. Problem Solving Systematics

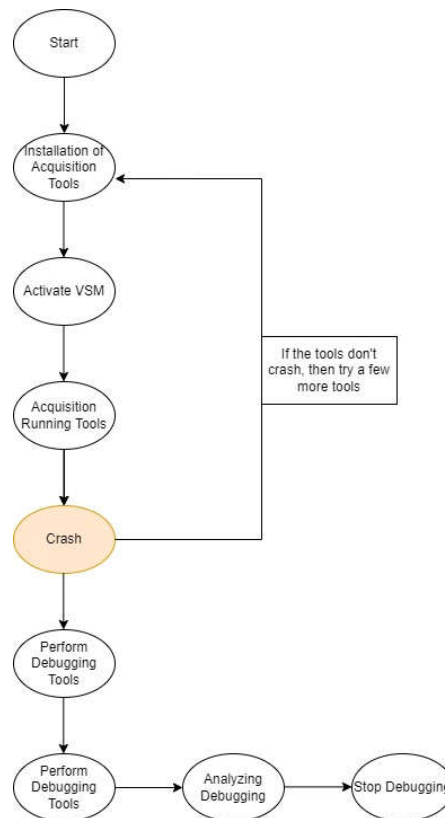
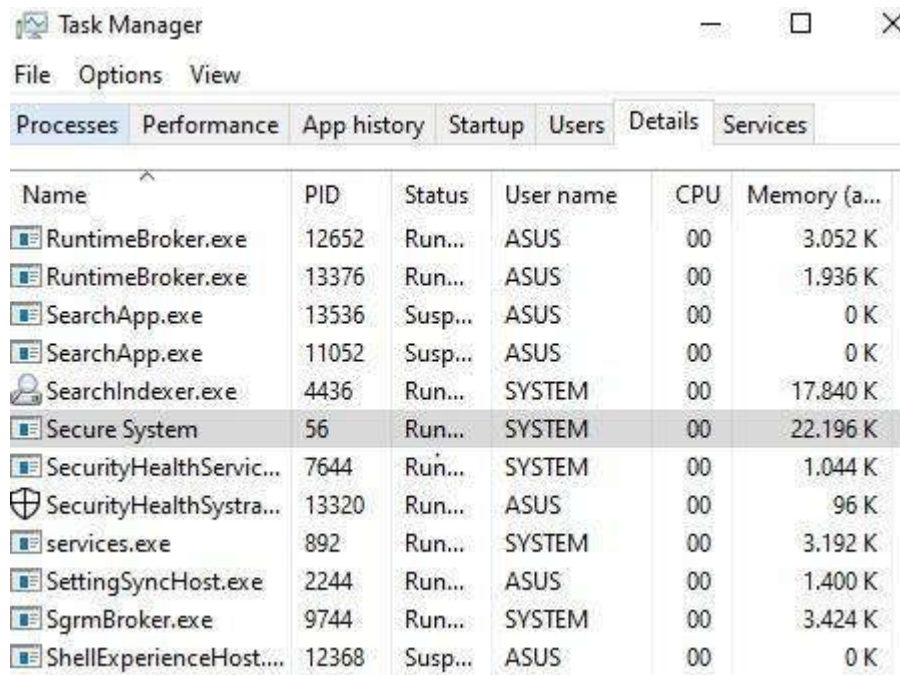


Figure 2 : Flowchart Overview of system design

In figure 2, the stages of research in the systematics of solving this problem use qualitative methods, namely literature studies which aim to collect more specific information related to the problem being studied, then this information will of course be utilized if it has something to do with the research being carried out which is shown by the theories relevant theory.

In this research, the method used to analyze the effect of VSM on the memory acquisition tool is dynamic code using WindBbg as its auxiliary tool. Software-based memory acquisition tools, namely autopsy, isobuster, DumpIt, Magnet RAM Capturer. However, using this tool on a system with active VSM mode causes a system crash known as a blue screen on death (BSoD). The following is proof that VSM is active by looking for "Task Manager" and checking whether "Secure System" is running or not can be seen in figure 3.



Name	PID	Status	User name	CPU	Memory (a...
RuntimeBroker.exe	12652	Run...	ASUS	00	3.052 K
RuntimeBroker.exe	13376	Run...	ASUS	00	1.936 K
SearchApp.exe	13536	Susp...	ASUS	00	0 K
SearchApp.exe	11052	Susp...	ASUS	00	0 K
SearchIndexer.exe	4436	Run...	SYSTEM	00	17.840 K
Secure System	56	Run...	SYSTEM	00	22.196 K
SecurityHealthServic...	7644	Run...	SYSTEM	00	1.044 K
SecurityHealthSystra...	13320	Run...	ASUS	00	96 K
services.exe	892	Run...	SYSTEM	00	3.192 K
SettingSyncHost.exe	2244	Run...	ASUS	00	1.400 K
SgrmBroker.exe	9744	Run...	SYSTEM	00	3.424 K
ShellExperienceHost...	12368	Susp...	ASUS	00	0 K

Figure 3 : Active VSM Task Manager

Computer software is prone to many flaws and problems. Blue screen of death (BSOD) and timeout detection and recovery (TDR) are the most common errors in computer software. Debugging is one of the most important things in the modern world to find and fix these errors. Debugging is part of the software testing process and depends on it throughout the software development life cycle. "WinDbg" is the most commonly used tool for debugging. This document provides basic debugging, TDR / BSOD debugging, and suggestions for new tools that can analyze crash dumps. This document proposes a tool that can use dumps generated during TDR/BSOD to get pre-debug information for each error. This tool analyzes this crash dump without actually debugging it on the target system can be seen in figure 4.

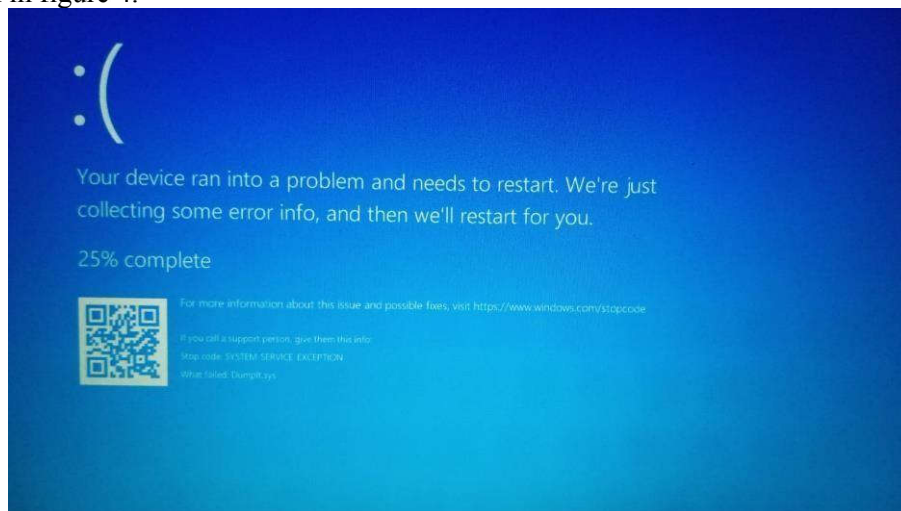


Figure 4: Memory acquisition results when VSM is active

B. Experiment Scenario

In this experiment, several stages of execution will be carried out. The test variables are several factors that cause the system to crash. Meanwhile, the test parameters are used to determine system performance when observed. the debugger can show memory address values as they change during program execution. In this test the test variables used are:

- Enable VSM
 1. UEFI Secure Boot is enabled.
 2. The Hyper-V role must be installed in Windows 10.

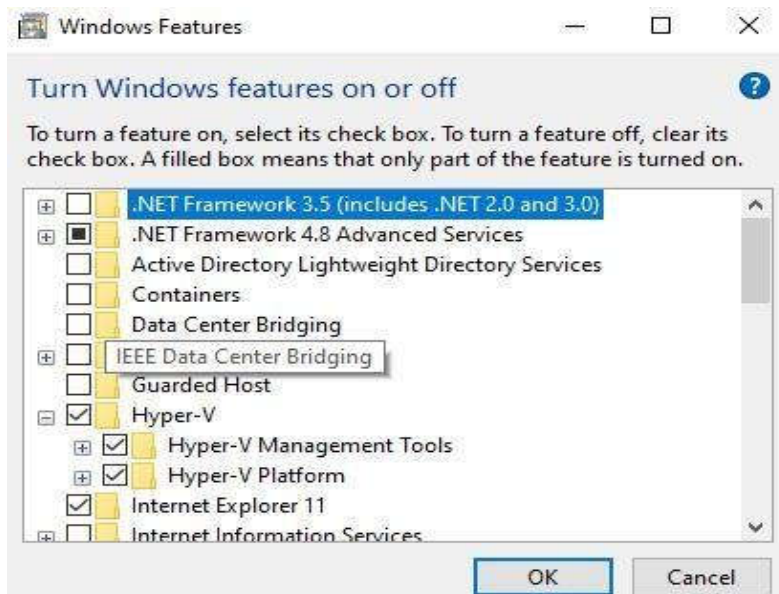


Figure 5: Hyper-V active

3. Virtual Secure Mode (VSM) must be enabled in Computer Configuration -> Administrative templates -> System -> Device Guard -> Turn on Virtualization Based Security. Enable secure boot then select Platform security level check Enable Credential Guard (LSA isolate)

4. VSM condition [enabled/disabled]

The VSM feature is activated via the BIOS by setting the "Intel Virtualization Technology" option to "Enabled". The BIOS used is the output of the BIOS Version/Date vendor American Megatrends Inc. X407UA.301, 12/03/2018.

- Checking the occurrence of crush

1. Install memory acquisition tools DumpIt, Memory RAM Capture, WindBG.
2. Perform live memory acquisition can be seen in figure 6.

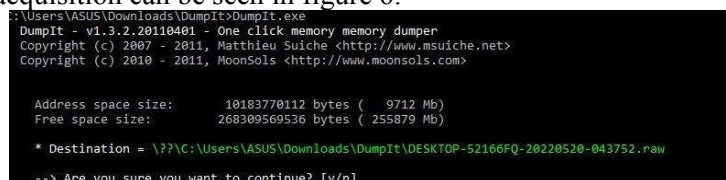


Figure 6 : Acquisition

3. Click yes, then the system will experience a BSOD

C. Application Debugging

The following are the steps for debugging:

1. Monitoring with Process Monitor

First delete all currently fetched events to remove irrelevant data by selecting Edit Clear Display. Next, run the subject malware with catch turned on. After a few minutes, you can stop recording the event.

2. Viewing Processes with Process Explore

Process Explorer monitors the processes running on the system and shows them in the displays child and parent relationships tree.

3. Step Code with Single-Stepping

Single-Stepping i.e. one step through the running program then returns control to the debugger. One step can see everything that is happening in a program.

4. Pausing Execution with Breakpoints

Breakpoints are used to pause execution and allow to check program status. When a program is paused at a breakpoint, it is called broken. Breakpoints are needed because they cannot access registers or memory addresses while the program is running, because these values are constantly changing.

5. Execution Modification with Debugger

The debugger can be used to modify program execution. Instructions, or the code itself to change the way that a program is executed. For example, to escape function calls, by setting a breakpoint where the function is called. When the breakpoint is hit, it can set the instruction pointer to the after-call instruction, thereby preventing the call

from taking place. If the function is critical, the program may not run properly when skipped or may crash. If its function has no impact on other areas of the program, the program may continue to run without problems.

6. Kernel Debugging with WindBg

If the virtual machine is running, the debugger should connect within a few seconds. If not running, the debugger will wait for the OS to boot, and then connect during the boot process. Once the debugger is connected, consider enabling verbose output while debugging the kernel, so that you get a more complete picture of what's going on with the verbose output, notified each time the driver unloads.

The results of this evaluation are conducting research on live memory acquisition using tools namely autopsy, isobuster, DumpIt, Magnet RAM Capturer. Which aims to understand what causes the system to crash.

III. RESULTS AND DISCUSSION

A. The Results

There are four tools used when performing memory acquisition, namely autopsy, isobuster, DumpIt, Magnet RAM Capturer. The acquisition tools that managed to experience a crash when VSM was active were the DumpIt tools because the system experienced a Blue Screen of Death (BSOD) and therefore there was no memory dump that could be executed. Meanwhile, the tools that have succeeded in acquiring memory are autopsy, isobuster and Magnet RAM Capturer. Here, dynamic code analysis is performed on a DumpIt application with an active VSM environment, and we want to know if the application is related to the BSOD error that occurs.

To get information on the "Bugcheck" event, you can use the Windbg application and run the !analyze -v command. The results can be seen in figure 7 and figure 8:

```
z: kd> !analyze -v
*****
*
*                               Bugcheck Analysis
*
*****

SYSTEM_SERVICE_EXCEPTION (3b)
An exception happened while executing a system service routine.
Arguments:
Arg1: 00000000c0000005, Exception code that caused the BugCheck
Arg2: fffff802459988a0, Address of the instruction which caused the BugCheck
Arg3: fffffd08127a0b00, Address of the context record for the exception that caused the BugCheck
Arg4: 0000000000000000, zero.

Debugging Details:
-----

Unable to load image \??\C:\WINDOWS\SysWOW64\Drivers\DumpIt.sys, Win32 error 0n2

KEY_VALUES_STRING: 1

    Key : Analysis.CPU.mSec
    Value: 8093

    Key : Analysis.DebugAnalysisManager
    Value: Create

    Key : Analysis.Elapsed.mSec
    Value: 14161

    Key : Analysis.IO.Other.Mb
    Value: 0

    Key : Analysis.IO.Read.Mb
    Value: 0

    Key : Analysis.IO.Write.Mb
    Value: 2

    Key : Analysis.Init.CPU.mSec
    Value: 2859

    Key : Analysis.Init.Elapsed.mSec
    Value: 53924

    Key : Analysis.Memory.CommitPeak.Mb
    Value: 95
```

Figure 7: !analyze -v

```

PROCESS_NAME: DumpIt.exe

STACK_TEXT:
ffffd008`127a1508 fffff802`45992d6f : 00000000`00000000 fffff888`b51aac80 00000000`00000000 fffff802`2013c490 : DumpIt+0x88a0
ffffd008`127a1510 fffff802`45991f0a : 00000000`00100000 00000002`00000001 00000000`03000000 fffff9f8`fc380000 : DumpIt+0x2d6f
ffffd008`127a1ac0 fffff802`459915b5 : fffff888`a9bfd5d0 00000000`00000001 00000000`c0000001 fffff888`00000001 : DumpIt+0x1f0a
ffffd008`127a1d00 fffff802`1e82a6b5 : fffff888`b34a2420 fffff888`b34a2538 00000000`00000002 00000000`000003b0 : DumpIt+0x155b
ffffd008`127a2700 fffff802`1ec164c8 : fffffd08`127a2a80 fffff888`b34a2420 00000000`00000001 fffff888`b47ba080 : nt!IofCallDriver+0x55
ffffd008`127a2740 fffff802`1ec162c7 : 00000000`00000000 fffffd08`127a2a80 00000000`00000005 fffffd08`127a2a80 : nt!IoPynchronousServiceTail+0x1a8
ffffd008`127a27e0 fffff802`1ec15646 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!IoPxxxControlFile+0xc67
ffffd008`127a2920 fffff802`1ea0aab5 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!NtDeviceIoControlFile+0x56
ffffd008`127a2990 00000000`77961cfc : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KISystemServiceCopyEnd+0x25
00000000`009be918 00000000`00000000 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : 0x77961cfc

SYMBOL_NAME: DumpIt+88a0
MODULE_NAME: DumpIt
IMAGE_NAME: DumpIt.sys
STACK_COMMAND: .cxr 0xffffdd08127a0b00 ; kb
BUCKET_ID_FUNC_OFFSET: 88a0
FAILURE_BUCKET_ID: AV_DumpIt!unknwon_function
OS_VERSION: 10.0.19041.1
BUILDLAB_STR: vb_release
OSPLATFORM_TYPE: x64
OSNAME: Windows 10
FAILURE_ID_HASH: {d498a4a8-6cc2-6768-9623-3766d4a295e6}
    
```

Figure 8 : Bugcheck analysis

Bugcheck analysis in the image above that executes the MEMORY.DMP file that crashes related to the DumpIt application. The module obtained is DumpIt, and the image_name obtained is DumpIt.sys. This causes information on the BSOD (Blue Screen Of Death) screen, or is called a crash. Windbg points out that DumpIt.sys is related to DumpIt can be seen in table 1.

TABEL I
 EXECUTABLE SEARCH PATH :

Child-SP	RetAddr	Call Site
00070000	000a5000	C:\Users\ASUS\Downloads\DumpIt\DumpIt.exe
57ed0000	58028000	C:\Users\ASUS\AppData\Local\Microsoft\WindowsApps\Microsoft.Windows.Common-Desktop-Background\TD\wow64\TTDRecordCPU.dll
76100000	7631c000	C:\WINDOWS\System32\KERNELBASE.dll
76710000	7678b000	C:\WINDOWS\System32\ADVAPI32.dll
768d0000	769c0000	C:\WINDOWS\System32\KERNEL32.DLL
769c0000	76a7e000	C:\WINDOWS\System32\RPCRT4.dll
77140000	771ff000	C:\WINDOWS\System32\msvrt.dll
77200000	77276000	C:\WINDOWS\System32\sechost.dll
774e0000	77525000	C:\WINDOWS\System32\SHLWAPI.dll
77840000	779e4000	C:\WINDOWS\SYSTEM32\ntdll.dll

From the results of the search path executable table, the file C:\WINDOWS\SYSTEM32\ntdll.dll is obtained. Where the file gets a syscall breakpoint which is located in the ntdll.dll file, the file has an error in the Operating System where the file cannot be executed because it crashes which causes the BSOD (Blue Screen Of Death) screen.

The code that caused the system to crash, namely code 80000003, experienced a Break Instruction Exception, with the name failure bucket ID being BREAKPOIN 80000003 ntdll.dll!LdeInitializeThunk.

B. Analysis of Test Results

Analysis of this dynamic code performs experimental results using four memory acquisition tools that are performed during VSM (Virtual Secure Mode), namely autopsy, isobuster, DumpIt, Magnet RAM Capturer. By using windbg this can help acquire memory in order to get a syscall breakpoint before the tool crashes. The next step is to help carry out the analysis using DR.Memory

"C:\ProgramFiles(x86)\Dr.Memory\bin64\dimemory.exe""C:\Users\ASUS\Downloads\Dumpit\DumpIt.exe". The main purpose of using DR.Memory is to look for syscall breakpoints when the VSM is active.

DR.Memory managed to access the module "C:\WINDOWS\system32\ntdll.dll" after analyzing the module to make a breakpoint on the DumpIt tools. DR.Memory is only able to see the last breakpoint module, it is necessary to do a test analysis using dynamic code, namely Windbg. This aims to determine whether the last module in DR.Memory is the same as Windbg.

In Windbg dynamic code analysis can be executed by the operating system. This analysis can be obtained by following the steps, namely on the windbg preview menu we can select the "Start debugging" menu, then select "Launch Executable (Advanced)" from the DumpIt application, and the debugger downloads the "wntdll.pdb" symbol file. After that, on the "Command" page, do the following command:

- To load symbols:
 1. .symfix
 2. .reload
 3. !analyze -v
- To run the DumpIt application
 1. g

From this analysis it can be seen that the line code before the crash on the DumpIt application was code 80000003, and the last module accessed on the DumpIt tools read by Windbg was C:\WINDOWS\SYSTEM32\ntdll.dll. The line code before the crash can mean that this error is caused by some conflicting Registry files, this is due to missing drivers or related to incompatible hardware on which the program is running. This is because it can't process JIT_DEBUG_INFO, Win32 error 0n30, it's an error in missing driver i.e. C:\Users\ASUS\Downloads\DumpIt\DumpIt.exe;C:\MyProjects\DisplayGreeting\Debug, the file doesn't have the correct path more specific so that it causes a breakpoint on line code 80000003.

The effect that is obtained when Virtual Secure Mode (VSM) in the dynamic memory acquisition process is that it causes the screen to experience a BSOD when performing memory acquisition, so this is the effect that is obtained when VSM is active. The problem that was obtained because of VSM was a crash on the ntdll.dll module which was caused by a missing registry file in the Bug check analysis of this dump file showing the cause of the system experiencing BSOD, namely when executing the dumpIt.sys module

The results obtained for this study were able to find out what caused the system to crash and which code was experiencing BSOD compared to previous research only obtaining the ad_driver.sys module. Where the previous research carried out dynamic analysis using the windbg application that was executed, namely the FTK Imager, while in my research the application that was executed was DumpIt. In previous research, the last module obtained before the system crash was C:\Windows\system32\mssprxy.dll, while my research obtained the last module, namely C:\WINDOWS\SYSTEM32\ntdll.dll. So, my research was able to find the last code before the crash was located at a breakpoint in line code 80000003 which was caused by a file that didn't have a more specific path.

IV. CONCLUSION

The results of dynamic code analysis using the Windbg acquisition application are carried out when VSM is active. What is done to carry out live memory acquisition is that there are four tools used when carrying out memory acquisition, namely autopsy, isobuster, DumpIt, Magnet RAM Capturer. The tools that have successfully performed memory acquisition are autopsy, isobuster and Magnet RAM Capturer. While the tools that failed to do memory-acquisition, namely DumpIt, this was caused by several errors that caused a crash due to a Break Instruction Exception, with the name failure bucket ID being BREAKPOINT_80000003_ntdll.dll!LdeInitializeThunk. In this line code, it can be seen that the cause of the crash is located in an error caused by a registry file that is contradictory to execution and related to incompatible

hardware. Whereas the last module accessed by Windbg was C:\WINDOWS\SYSTEM32\ntdll.dll, the file had an error in the Operating System where the file could not be executed because it crashed which caused the BSOD (Blue Screen Of Death) screen.

VSM is Windows 10 which is used to make managing the existing environment on the operating system to be safe, VSM itself is separate from the usual Windows environment. How the dumpIt tool works is a combination of win32dd and win64dd, combined into one executable. DumpIt will then take a snapshot of the host's physical memory and save it to the folder where the DumpIt executable resides. The impact caused by active VSM when conducting experiments on the Windbg application to acquire memory there is a crash located in the operating system which causes a BSOD (Blue Screen Of Death).

REFERENCES

- [1]Parmaza , B. (2018). Apa itu Digital Forensics (Forensik Digital). *Komunitas Teknologi dan Komunikasi Jambi*.
- [2]Aleksandar Milenkoski, D. P. (2021). *Virtual Secure Mode: Architecture Overview*. [Technical Report] ERNW Enno Rey Netzwerke GmbH.
- [3]Anand, G. (2021). Windbg A-Complete Gide For Advanced Windows Debugging. *windbg a complete guide*.
- [4]Arwidmark, J. (2015). Virtual Secure Mode (VSM) explained. *Enabling Virtual Secure Mode (VSM) in Windows 10 Enterprise Build 10130*.
- [5]George, G., & Inani, S. (2018). *Learning Malware Analysis*. Packt Publishing Ltd. Livery Place 35 Livery Street Birmingham B3 2PB, UK.
- [6]Irfan Febrian Editia Kurdiat, N. W. (2016). Analisis Proses Investigasi Dekstop PC Yang Terhubung Layanan Private Cloud. *Jurnal Teknik Informatika dan Sistem Informasi*.
- [7]Michael Solomon, D. B. (2005). *Computer Forensics Jumpstart*. Alameda: SYBEX Inc .
- [8]Rahevar, D. (2013). Study on Live analysis of Windows Physical Memory,IOSR Journal of Computer Engineering (IOSR-JCE). *IOSR Journal of Computer Engineering (IOSR-JCE)*.
- [9]Riadi, I., Fadlil, A., & Hafizh, M. N. (2020). Analisis Bukti Serangan Address Resolution Protocol Spoofing menggunakan Metode National Institute of Standard Technology. *Jurnal Pendidikan Informatika*.
- [10]Umar, R., Riadi, I., & Handoyo, E. (2019). Analisis Keamanan Sistem Informasi Berdasarkan Framework COBIT 5 Menggunakan Capability Maturity Model Integration (CMMI). *Jurnal Sistem Informasi Bisnis*.
- [11]Cahyani, N. D., Jaded, E. M., & Ab Rahman, N. H. (2022). The Influence of Virtual Secure Mode (VSM) on Memory Acquisition. *International Journal of Advanced Computer*.