ISSN 2460-9056
http://socj.telkomuniversity.ac.id/ijoict/

Intl. Journal on ICT
Vol. 9, No. 1, Jun 2023. pp. 1-13
doi: doi.org/10.21108/ijoict.v9i1.688

Static Code Analysis on the Effect of Virtual Secure Mode on Memory Acquisition with IDA

Nadja Shafa Adryana[1], Niken Dwi Wahyu Cahyani[2], Erwid Musthofa Jadied[3]

[123]*School of Computing, Telkom University*
1 Telekomunikasi St., Terusan Buahbatu, Bandung, West Java, Indonesia 40257

[1]nadjashafa@student.telkomuniversity.ac.id, [2]nikencahyani@telkomuniversity.ac.id,
[3]jadied@telkomuniversity.ac.id

Abstract
Memory acquisition process is one of digital forensics act. There are several tools that support memory acquisition process. At this time, there is a feature named secure mode that can caused crash or error in memory acquisition tools system and caused the tools to be unusable, also the loss of the computer memory. This study is experimenting to find the effect on memory acquisition tools performance while running in secure mode. After getting the experiment results, the analysis is going to be carried out towards memory acquisition tools using static code analysis, which is one of the techniques of reverse engineering, using IDA. This study aims to find any kind of occurrences that happen on memory acquisition process while in secure mode and find the cause of it. The purpose of this study is to be useful for digital forensic tester in understanding the potential risk of the secure mode impact in acquisition process. The experiment shows that Autopsy version 4.7 cannot run properly in VSM environment, different with FTK Imager. The results from the analysis define that the difference between library on normal kernel and secure kernel is the one that caused the program to terminate while in secure mode. In advance, the operating system that runs in the device are the other reasons of memory acquisition tools cannot run properly on VSM environment. It is caused by the difference in security features that is being provided by a specific operating system.

Keywords: digital forensic, static analysis, virtual secure mode, memory acquisition tools

I. INTRODUCTION

In criminal cases, digital forensics is required to investigate digital evidence. This digital evidence can be obtained from a computer which later are going through the memory acquisition process [1]. Each computer has different operating system and different security system. Regarding the security system on computers, on Windows-based devices, Microsoft implements a Virtual Secure Mode (VSM) system. VSM is a feature in Windows 10 that is used to secure the operating system. Furthermore, VSM is a secure environment that functions to protect the operating system from attack targets [2]. In this condition, user is going to enter the IUM (Isolated User Mode) stage which is going to isolate the user into a separate environment from Windows main kernel.

Received on 18 Jan 2023. Revised on 30 Jan 2023. Accepted and Published on 09 Jun 2023.

IUM works directly with other features in Windows 10, Credential Guard, and Device Guard. Credential Guard uses IUM as a defense against hash attacks [3]. IUM relies on a secure kernel to handle the work of the base operating system. When secure kernel handles this, the kernel on Windows is the one that handle the main operations running on the device. The main advantage of using a separate kernel to handle IUM is that the secure kernel does not contain third party modules. This allows the secure kernel to operate without the risk of interference from third party code. To achieve IUM, the secure kernel relies on the Windows hypervisor, Hyper-V, to manage both the Windows kernel and the secure kernel. In other word, the hypervisor can set different memory permissions for the two kernels, so that it is impossible for a normal kernel to access the secure kernel's memory. Hyper-V holds the root partition. In this partition there are two kernels and two types of user mode. This isolation is implemented by Hyper-V as the basic entity that manages the execution of the environment within VSM [2].

One of digital forensics act that is frequently done is obtaining digital evidence from a device. The action is required to use some tools to restore the memory that has been deleted, called memory acquisition tools. There are several memory acquisitions tools that can be used by digital forensic. Normally, these tools are being run on normal kernel, not on the secure kernel. Regarding to the explanation above, secure mode runs on a different kernel separated with the main kernel of Windows. This is the reasons of this study being done. This study is helping digital forensics to prevent any kind of issues that comes up while performing memory acquisition process in further time. This study is going to figure out whether it has differences while performing memory acquisition process on secure mode and normal mode, or not. Furthermore, figuring out what can be the cause of the differences. This study is being carried out with static code analysis, after the experiment being done on different environment of Windows.

Static code analysis is one of a reverse engineering process to verify the executable file or usually known as .exe file regardless to view all the instruction from the program. One of benefits of this method, it provides the information about significant function of the program, and in some cases could disclose whether the file has some issues or not. This method can evaluate the whole source code in more efficient time using some tools, different with manual code analysis. Other than that, this method can be done without any needs of online networking [4], [5]. Static code analysis can be done using several methods, some of them are antivirus scanning, finding strings, runtime system analysis, disassembling, and so on. Static code analysis can be done using many forensics tools. One of the famous tools is Interactive Disassembler (IDA). This tool is being used for performing forensics act because it is not only performing code disassembling, but more act such as identifying error, finding connection between several functions, and analyzing vulnerabilities.

Based on the background above, this study is experimenting memory acquisition process being run on different Windows environment, secure mode, and normal mode. The memory acquisition tools that being used in this study are FTK Imager and Autopsy. In this case, memory acquisition tools are going to be testified with and without VSM running on Windows 10 operating system. After getting the experiment results and finding some issues while running the tools, this leads to performing static code analysis towards memory acquisition tools using finding strings methods with the help of Interactive Disassembler (IDA) and using runtime system analysis methods directly on Windows operating system itself.

Research related to this study is being done by Niken [6]. The research shows that FTK Imager cannot run properly on the device being used. From Niken’s research, it refers that the author is suggesting further study about related problem but with other various variables, for more prominent results. This study is continuing the previous research to find any new variables that coming up while acquiring memory in secure mode with a few differences. The differences are, this study is explaining on how to enter VSM in details, this study is also going to testify the same tools which is FTK Imager, but with different device that has different specification inside. Furthermore, this study is adding a different tool from the previous research, which is Autopsy. The purpose of using different tool is to identify whether this tool is being affected by the secure mode, or not.

The occurs upon device that being used is going to be the starting point to this study. The problem that appears is the reason to analyze the memory acquisition tools whether the tools source code have issues that not supporting the application for running on secure mode, or it is the Windows system that caused the interruption on the memory acquisition tools performance. The final output of this study is knowing whether the secure mode is affecting the memory acquisition process or not, and what caused the secure mode to affect its process.

II. LITERATURE REVIEW

Virtual Secure Mode (VSM) is a feature in Windows 10 that can put the user in two conditions, namely normal mode, and secure mode. This secure mode is called Isolated User Mode (IUM) that set up a safe and isolated operating system environment from the usual Windows environment. This feature is designed to protect the operating system from attacks [2], [3]. In Windows 10, there is a hypervisor system called Hyper-V. Processes that should be running on windows, are moved to separate hardware-based Hyper-V containers, which are inaccessible to Windows. If the kernel on Windows is compromised, the processes stored on VSM must remain safe.

Researched by Hans Kristian in 2017, shows how to apply reverse engineering, and it reveals the analysis result from secure mode in Windows 10. The purpose of using this research as a reference is that this journal explains the basic of secure mode on Windows 10, also explain reverse engineering process that being carried out in this study [7].

Related to digital forensic act, M. Azhar [1] doing researched about data acquisition and vanishing technique. This journal explains how to carry out data vanishing after using memory acquisition tools, FTK Imager. Other than that, this research shows how to do the code analysis while doing data vanishing, which later being applied for this study with VSM condition is turned on.

For static code analysis, in Michael Sikorski’s book [4], he talks about every step for doing malware analysis, both static and dynamic code analysis. This book breaks down the code analysis in more detail for readers to understand easily in general explanation. A few techniques such as antivirus scanning, finding strings, and runtime analysis are being explained as well in this book. For the advanced static analysis, IDA is one of the tools that help to execute the process. This tool is also being explained in the book about how to operate it. This shows that IDA is a recommended tools to use for doing static code analysis because it performs as a powerful disassembler for doing digital forensics act.

Doing static code analysis, especially in operating IDA, needs some advance knowledge in system architecture, particularly in understanding the instruction on how the device work. Heru [8], explain about commands that being analyzed while performed code analysis using IDA. The commands are showing the functions that the device is running while executing the program, so the program can run properly. Furthermore, the researched journal is used as reference for this study because in static code analysis, knowing about every parameter to be examined is a must so the study is not going any further out of the parameters being set.

Meanwhile, Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna conducted research related to leak detection in iOS-based applications using IDA as a disassembler in the CFG (Control Flow Graphs) extraction process. But in this case, IDA does not complete its call target to the sending function. IDA only recognizes calls to the dynamic function itself and cannot directly detect leaks to the application [9].

This study is continuing the research being done by Niken [6]. The purpose is for identifying any new variables that comes up to the related issues. The parameters are the difference between the methods, the memory acquisition tools, and the device that being used.

III. RESEARCH METHOD

The general design process of this study is shown in Fig. 1.

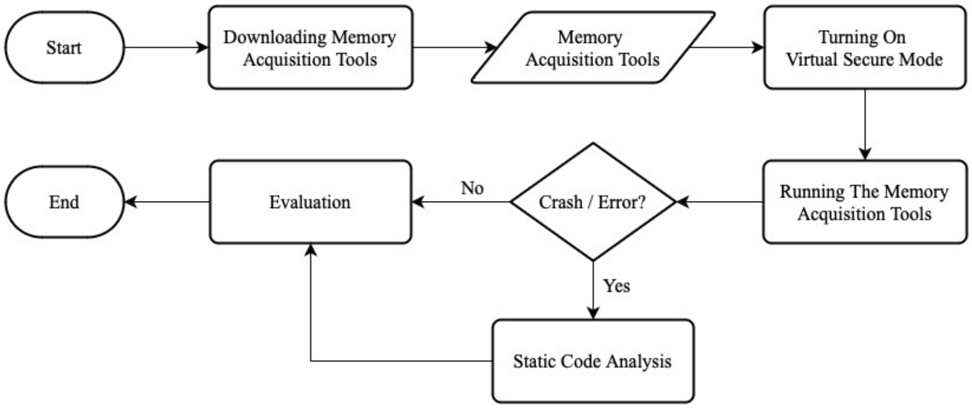

Start
Downloading Memory Acquisition Tools
Memory Acquisition Tools
Turning On Virtual Secure Mode
Running The Memory Acquisition Tools
Crash / Error?
No
Yes
Evaluation
End
Static Code Analysis

Fig 1. Flowchart of the general process

A. Downloading Memory Acquisition Tools on Device

The act of creating a set of parameters for an intricate hardware system by a systematic analysis of the system is known as reverse engineering, which can be helpful for maintaining the system [10], [11]. As static code analysis is one of the reverse engineering processes, it requires to know exactly every detail needed for this experiment, from the device that being used to every tool that being examined. This study is using two memory acquisition tools, each with different version. Details about the device is shown in Table 1, and the memory acquisition tools is shown in Table 2.

TABLE I
DEVICE SPECIFICATION

Name	Specification
Device	HP Elitebook 820 G4
Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz (4 CPUs), ~ 2.7GHz
Memory	8192MB RAM
BIOS	P78 Ver. 01.16
Operating System	Windows 10 Pro 64-bit (10.0, Build 19044)

TABLE 2
MEMORY ACQUISITION TOOLS

Tools	Version	Download Source
FTK Imager	4.5.0.3	https://accessdata.com/product-download/ftk-imager-version-4-5
FTK Imager	4.3.0.18	https://accessdata.com/product-download/ftk-imager-version-4-3-0
Autopsy	4.19.3	https://www.autopsy.com/download/
Autopsy	4.7.0	https://github.com/sleuthkit/autopsy/releases?page=2

After downloading tools from Table 2, the next step is checking whether all the tools are running properly on the device with VSM turned off.

B. *Turning On Virtual Secure Mode*

Before starting on this step, make sure that the operating system that runs in the device is Windows 10 based. The next phase for entering secure mode, or so-called safe mode, is being explained down below.

1) *Ensuring Hyper-V is Turned On*
Before entering safe mode, make sure Hyper-V is turned on by opening "Apps and Features". Then, choose "Programs and Features" menu, and "Turn Windows Feature on or off". Find "Hyper-V" and if the all the checkboxes are already filled, then it can be continued to the next step. If not, click on all the checkboxes in "Hyper-V" selection, and restart the device. After restarting the device, double check the Hyper-V again before entering safe mode.

2) *Entering Safe Mode*
The following steps is one of the easiest ways to entering safe mode on Windows 10. First, is opening "Settings", then choose "Update and Security". On the sidebar, choose "Recovery" settings and click on "Restart now" selection on "Advanced startup" section. Wait around 10 seconds until the monitor displayed advance menu. Choose "Troubleshoot", then "Advanced option". After the advanced option menu appear, click on "Startup Settings", then click "Restart". There are several settings that can be chose in startup settings menu. For entering safe mode, choose between menu number 4 for safe mode only, or menu number 5 for safe mode with networking.

3) *Ensuring Secure System is Running*
In this part, after entering the safe mode, the desktop wallpaper is going to turned into black. Ensuring if the secure system is running can be done by checking on "Task Manager". If it runs, "Secure System" is shown and running on the device.

C. *Running The Memory Acquisition Tools*

On this phase, run the memory acquisition tools that already being downloaded before with VSM or safe mode condition is turned on. If it happens to be crashed or error on the application or device, eventually that memory acquisition tools is going to be analyzed on the next step.

D. *Static Code Analysis*

After finding out which tools that crashed or error during application testing while VSM turned on, static code analysis is the next step to be done. This process is going to be analyzing which part that caused the errors on memory acquisition tools. Here are some actions of static code analysis that being applied for this study.

1) *Disassembly*
One of reverse engineering technique is disassembly process. IDA is used to translate binary code or executable file of the program into assembly language. This process can be done in IDA by choosing "View" menu and click on "Open subview", then choose "Disassembly".

2) *Search String*
This process being carried out by tracing the code on the IDA by paying attention to the error strings in the memory acquisition tools. The purpose of the search string is to find errors in the functionality of the memory acquisition tools to be analyzed. This process can be done in IDA by tracing the strings that comes to have errors indication on the running process, by opening the occurrences of strings with error.

3) *Event Viewer Analysis*
Besides the actions above, this process is one of the methods that important to be applied in this study. From this analysis, we can view event log of the application that runs in the operating system. Event viewer is a feature provided in the operating system that can be accessed by searching on the search bar.

E. Testing Scenario

This study parameters are finding the cause of the program system to crash or error while VSM condition is turned on. Furthermore, finding on how VSM works and the impact of it on memory acquisition process. Other than the parameters, this study variables are VSM condition (on or off) and the difference between the memory acquisition tools that being used, shown in Table 2. Here are the testing scenarios that being done for this study.

1) Downloading memory acquisition tools from Table 2 and run them while VSM is turned off.
2) Restarting into BIOS on the device that being used in this study as defined on section B.
3) Running memory acquisition tools while VSM is turned on.
4) Opening executable file from certain memory acquisition tools.
5) Analyzing static code as defined on section D, regarding to the parameters and variables.
6) Saving and recording all the analysis results for the evaluation of this study.

F. Evaluation

Evaluation is carried out by presenting the results of static code analysis which caused the crashes and errors in memory acquisition tools. This evaluation aims to avoid crashes and errors when live forensics is being carried out with these memory acquisition tools later.

IV. RESULTS AND DISCUSSION

A. VSM Environment Testing

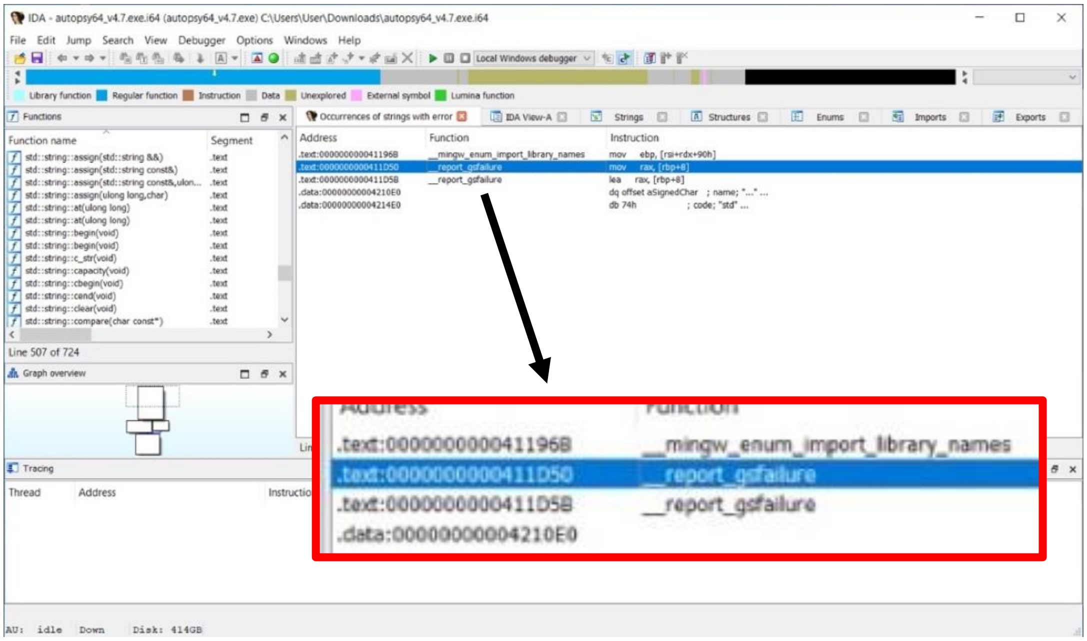
In this section, all memory acquisition tools that shown in Table 2 is being testify in different condition of VSM, turned on and turned off. The results of the experiment can be seen in Table 3. From Table 3, its being known that FTK Imager runs properly in the device with or without VSM being turned on. Further analysis is being done for this specific application in other section down below, comparing with another research about FTK Imager. Other than FTK Imager, Autopsy is being detected with errors coming up while VSM is turned on, but not for the latest version of Autopsy. The latest one, runs smoothly in VSM condition being turned on. It is expected that the difference between two Autopsy's source codes might be one of the causes for the errors in the old version of it.

TABLE 3
EXPERIMENT RESULTS ON DIFFERENT VSM ENVIRONMENT

Tools	Version	VSM Turned Off	VSM Turned Off
FTK Imager	4.5.0.3	Not crashed and no errors detected	Not crashed and no errors detected
FTK Imager	4.3.0.18	Not crashed and no errors detected	Not crashed and no errors detected
Autopsy	4.19.3	Not crashed and no errors detected	Not crashed and no errors detected
Autopsy	4.7.0	Not crashed and no errors detected	Application cannot be opened

B. *Static Code Analysis with IDA*

After going through testing experiment from the previous session, the application with errors detected, namely Autopsy, is going through the static code analysis to figure out what can be the cause of the errors. Firstly, Autopsy is going through disassembly process. This process is being executed with IDA. Fig. 2 display the occurrences of strings with errors window. This window shows strings that bring errors from the program execution.


IDA - autopsy64_v4.7.exe.i64 (autopsy64_v4.7.exe) C:\Users\User\Downloads\autopsy64_v4.7.exe.i64

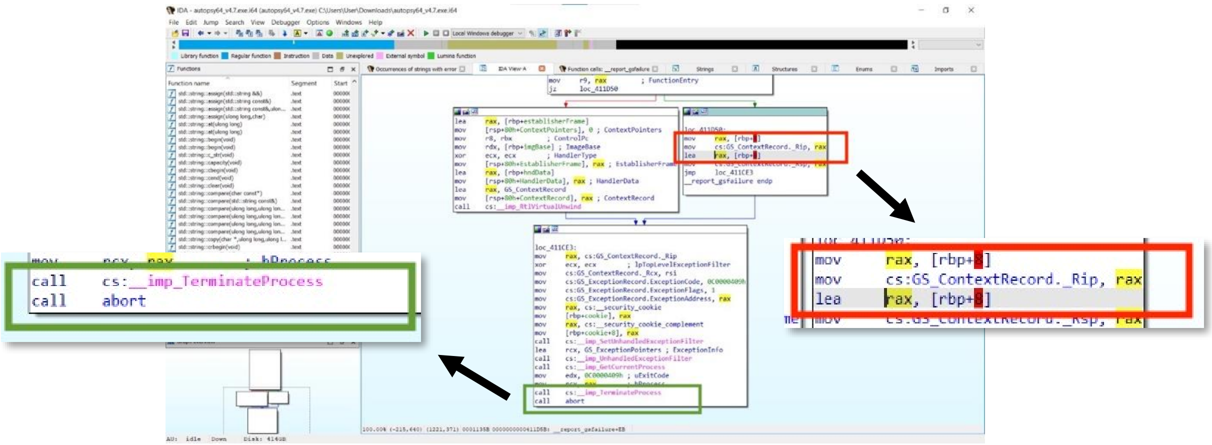
Function name	Segment
std::string::assign(std::string &&)	.text
std::string::assign(std::string const&)	.text
std::string::assign(std::string const&,ulon...	.text
std::string::assign(ulong long,char)	.text
std::string::at(ulong long)	.text
std::string::at(ulong long)	.text
std::string::begin(void)	.text
std::string::begin(void)	.text
std::string::c_str(void)	.text
std::string::capacity(void)	.text
std::string::cbegin(void)	.text
std::string::cend(void)	.text
std::string::clear(void)	.text
std::string::compare(char const*)	.text

Line 507 of 724

Address	Function	Instruction
.text:000000000041196B	__mingw_enum_import_library_names	mov ebp, [rsi+rdx+90h]
.text:0000000000411D50	__report_gsfailure	mov rax, [rbp+8]
.text:0000000000411D5B	__report_gsfailure	lea rax, [rbp+8]
.data:00000000004210E0		dq offset aSignedChar ; name; "..." ...
.data:00000000004214E0		db 74h ; code; "std" ...

Fig 2. __report_gsfailure function is being indicated with error

This indicates a function called __report_gsfailure which taking role as function that maintain the stack buffer overruns, is being expected as the problem that cause Autopsy to terminate immediately before even started in safe mode environment. The instruction being carried by this function is shown in Fig 3, inside the red box. The instruction of “MOV rax, [rbp+8]” and “LEA rax, [rbp+8]” indicates the error from the function. It is known that MOV is an instruction to move data to a different location. Furthermore, LEA (Load Effective Address) is an instruction to load the data. In this case, MOV is going to move the [rbp+8] to rax register, which is the register to return values in the function [12], and next being loaded in the register.


IDA - autopsy64_v4.7.exe.i64 (autopsy64_v4.7.exe) C:\Users\User\Downloads\autopsy64_v4.7.exe.i64
File Edit Jump Search View Debugger Options Windows Help
Local Windows debugger
Library function
Regular function
Instruction
Data
Unexplored
External symbol
Lumina function
Functions
Occurrences of strings with error
IDA View-A
Function calls: __report_gsfailure
Strings
Structures
Enums
Imports

```
mov     r9, rax         ; FunctionEntry
jz      loc_411D50
```

```
lea     rax, [rbp+establisherFrame]
mov     [rsp+80h+ContextPointers], 0 ; ContextPointers
mov     r8, rbx         ; ControlPc
mov     rdx, [rbp+imgBase] ; ImageBase
xor     ecx, ecx        ; HandlerType
mov     [rsp+80h+EstablisherFrame], rax ; EstablisherFrame
lea     rax, [rbp+hndData]
mov     [rsp+80h+HandlerData], rax ; HandlerData
lea     rax, GS_ContextRecord
mov     [rsp+80h+ContextRecord], rax ; ContextRecord
call    cs:__imp_RtlVirtualUnwind
```

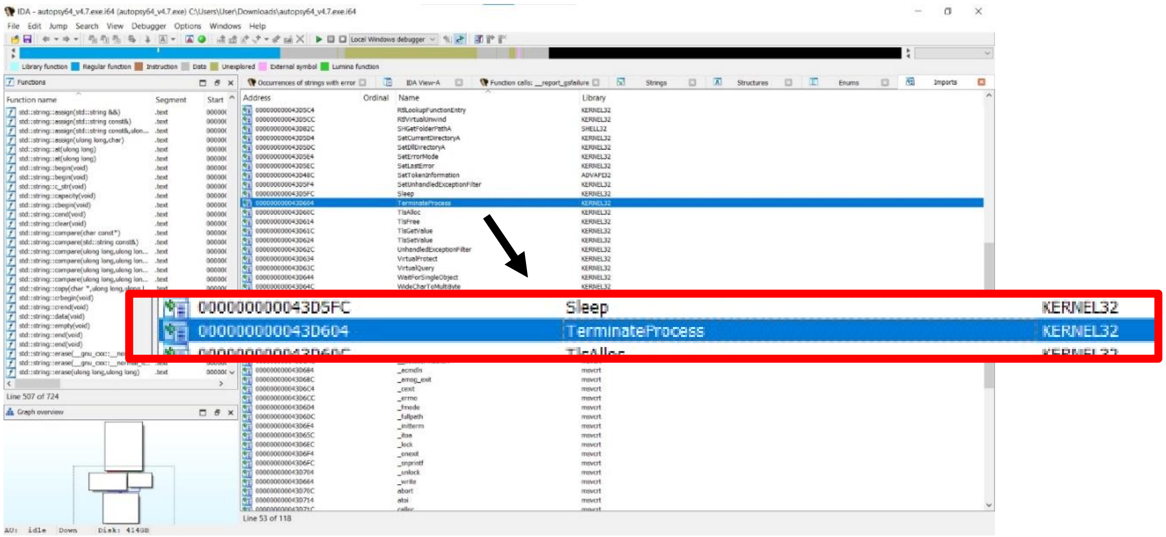
```
loc_411D50:
mov     rax, [rbp+8]
mov     cs:GS_ContextRecord._Rip, rax
lea     rax, [rbp+8]
mov     cs:GS_ContextRecord._Rsp, rax
jmp     loc_411CE3
__report_gsfailure endp
```

```
loc_411CE3:
mov     rax, cs:GS_ContextRecord._Rip
xor     ecx, ecx        ; lpTopLevelExceptionFilter
mov     cs:GS_ContextRecord._Rcx, rsi
mov     cs:GS_ExceptionRecord.ExceptionCode, 0C0000409h
mov     cs:GS_ExceptionRecord.ExceptionFlags, 1
mov     cs:GS_ExceptionRecord.ExceptionAddress, rax
mov     rax, cs:__security_cookie
mov     [rbp+cookie], rax
mov     rax, cs:__security_cookie_complement
mov     [rbp+cookie+8], rax
call    cs:__imp_SetUnhandledExceptionFilter
lea     rcx, GS_ExceptionPointers ; ExceptionInfo
call    cs:__imp_UnhandledExceptionFilter
call    cs:__imp_GetCurrentProcess
mov     edx, 0C0000409h ; uExitCode
mov     rcx, rax        ; hProcess
call    cs:__imp_TerminateProcess
call    abort
```

```
100.00% (-215,445) (1221,371) 0001135B 000000000411D5B: __report_gsfailure+EB
AU: idle   Down   Disk: 414GB
```

Fig 3. Graph from ___report_gsfailure function that indicates the error

Extending from the instruction, the impact of it is shown in Fig 3, inside the green box. It shows that the function calls two strings at the end, which are “TerminateProcess” and “abort”. These two strings are being imported from outside the application. “TerminateProcess” is being imported from KERNEL32 library, shown in Fig 4. And “abort” is being imported from mcvsrt library, shown in Fig 5.

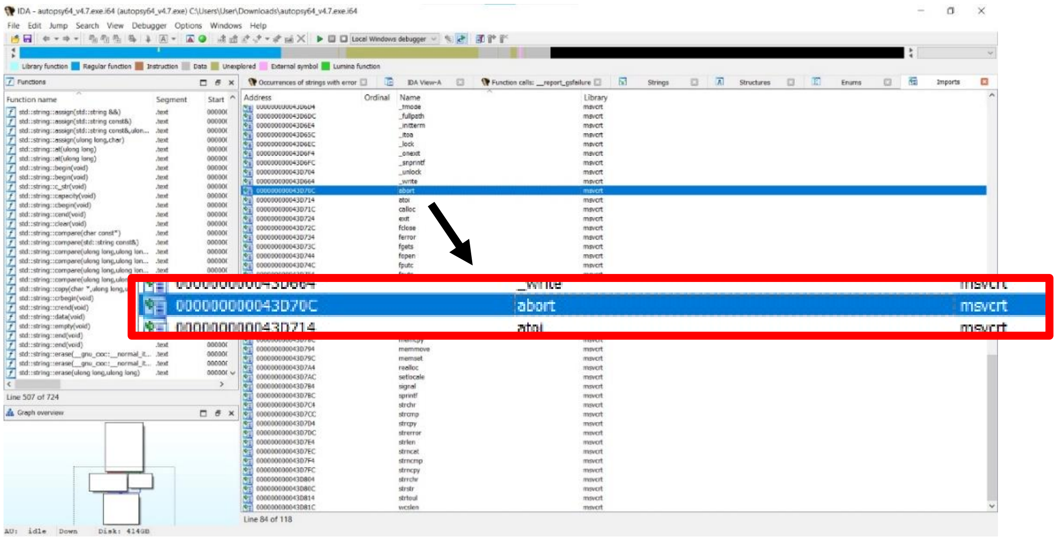

IDA - autopsy64_v4.7.exe.i64 (autopsy64_v4.7.exe) C:\Users\User\Downloads\autopsy64_v4.7.exe.i64
File Edit Jump Search View Debugger Options Windows Help
Local Windows debugger
Library function
Regular function
Instruction
Data
Unexplored
External symbol
Lumina function
Functions
Occurrences of strings with error
IDA View-A
Function calls: __report_gsfailure
Strings
Structures
Enums
Imports
Line 507 of 724
Graph overview
AU: idle Down Disk: 414GB

Address	Ordinal	Name	Library
000000000043D5C4		RtlLookupFunctionEntry	KERNEL32
000000000043D5CC		RtlVirtualUnwind	KERNEL32
000000000043D82C		SHGetFolderPathA	SHELL32
000000000043D5D4		SetCurrentDirectoryA	KERNEL32
000000000043D5DC		SetDllDirectoryA	KERNEL32
000000000043D5E4		SetErrorMode	KERNEL32
000000000043D5EC		SetLastError	KERNEL32
000000000043D48C		SetTokenInformation	ADVAPI32
000000000043D5F4		SetUnhandledExceptionFilter	KERNEL32
000000000043D5FC		Sleep	KERNEL32
000000000043D604		TerminateProcess	KERNEL32
000000000043D60C		TlsAlloc	KERNEL32
000000000043D614		TlsFree	KERNEL32
000000000043D61C		TlsGetValue	KERNEL32
000000000043D624		TlsSetValue	KERNEL32
000000000043D62C		UnhandledExceptionFilter	KERNEL32
000000000043D634		VirtualProtect	KERNEL32
000000000043D63C		VirtualQuery	KERNEL32
000000000043D644		WaitForSingleObject	KERNEL32
000000000043D64C		WideCharToMultiByte	KERNEL32
000000000043D684		_acmdln	msvcrt
000000000043D68C		_amsg_exit	msvcrt
000000000043D6C4		_cexit	msvcrt
000000000043D6CC		_errno	msvcrt
000000000043D6D4		_fmode	msvcrt
000000000043D6DC		_fullpath	msvcrt
000000000043D6E4		_initterm	msvcrt
000000000043D65C		_itoa	msvcrt
000000000043D6EC		_lock	msvcrt
000000000043D6F4		_onexit	msvcrt
000000000043D6FC		_snprintf	msvcrt
000000000043D704		_unlock	msvcrt
000000000043D664		_write	msvcrt
000000000043D70C		abort	msvcrt
000000000043D714		atoi	msvcrt
000000000043D71C		calloc	msvcrt

Line 53 of 118

Fig 4. TerminateProcess is being imported from KERNEL32

Msvcrt library is providing access to Windows which can be affecting the operating system [13]. KERNEL32 is a dynamic link library (.dll) which being used as the running operator of Windows [14]. The program cannot run properly and is being terminated by KERNEL32 because of the difference on the kernel base. While VSM environment being turned on, it runs on a different hypervisor which also has a separate kernel from the main Windows kernel.



```
IDA - autopsy64_v4.7.exe.i64 (autopsy64_v4.7.exe) C:\Users\User\Downloads\autopsy64_v4.7.exe.i64

Address              Ordinal  Name        Library
000000000043D6D4              _tmode      msvcrt
000000000043D6DC              _fullpath   msvcrt
000000000043D6E4              _initterm   msvcrt
000000000043D65C              _itoa       msvcrt
000000000043D6EC              _lock       msvcrt
000000000043D6F4              _onexit     msvcrt
000000000043D6FC              _snprintf   msvcrt
000000000043D704              _unlock     msvcrt
000000000043D664              _write      msvcrt
000000000043D70C              abort       msvcrt
000000000043D714              atoi        msvcrt
000000000043D71C              calloc      msvcrt
000000000043D724              exit        msvcrt
000000000043D72C              fclose      msvcrt
000000000043D734              ferror      msvcrt
000000000043D73C              fgets       msvcrt
000000000043D744              fopen       msvcrt
000000000043D74C              fputc       msvcrt
...
000000000043D794              memmove     msvcrt
000000000043D79C              memset      msvcrt
000000000043D7A4              realloc     msvcrt
000000000043D7AC              setlocale   msvcrt
000000000043D7B4              signal      msvcrt
000000000043D7BC              sprintf     msvcrt
000000000043D7C4              strchr      msvcrt
000000000043D7CC              strcmp      msvcrt
000000000043D7D4              strcpy      msvcrt
000000000043D7DC              strerror    msvcrt
000000000043D7E4              strlen      msvcrt
000000000043D7EC              strncat     msvcrt
000000000043D7F4              strncmp     msvcrt
000000000043D7FC              strncpy     msvcrt
000000000043D804              strrchr     msvcrt
000000000043D80C              strstr      msvcrt
000000000043D814              strtoul     msvcrt
000000000043D81C              wcslen      msvcrt
Line 84 of 118

000000000043D664              _write      msvcrt
000000000043D70C              abort       msvcrt
000000000043D714              atoi        msvcrt
```

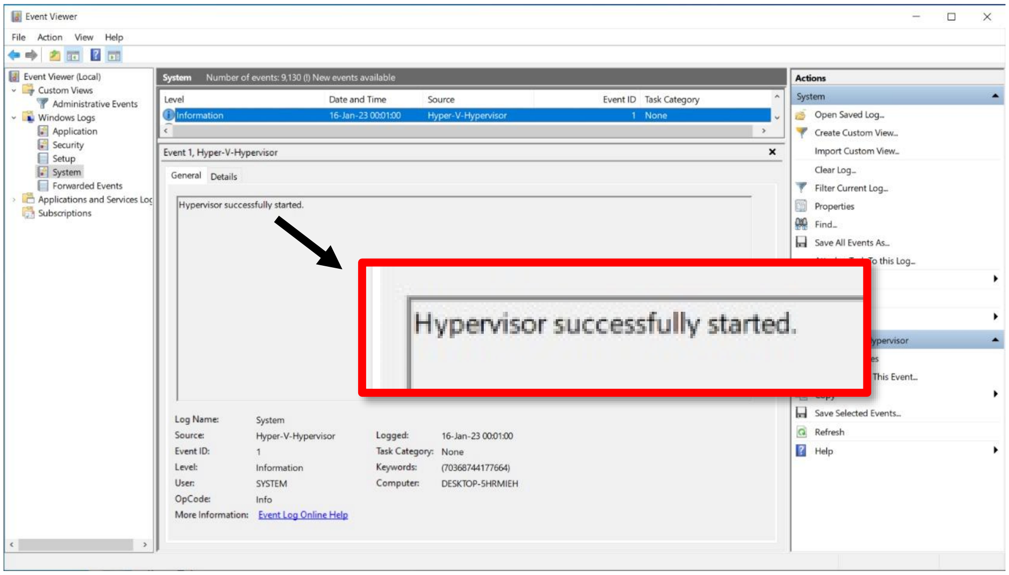
File Edit Jump Search View Debugger Options Windows Help
Local Windows debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
Occurrences of strings with error
IDA View-A
Function calls: __report_gsfailure
Strings
Structures
Enums
Imports
Function name Segment Start
Line 507 of 724
Graph overview

Fig 5. Abort is being imported from msvcrt

C. System Analysis

Every operating system has its own unique. This study is using Windows 10 which was launched in July 2015. From the previous version, which was Windows 8, there are not much difference in the system especially on the Event Viewer which being used in this analysis [15]. The specifications of the device and operating system being used in this study are already being shown on Table 1.

After running several experiments on the previous step, it is known that both versions of FTK Imager did not run into problem while being executed in VSM environment being turned on. This comes to the system analysis. This analysis is overcoming what kind of events that appear on the operating system after the experiment. Here are the events log sequence that appears after the experiment, shown in Fig 6 – 9.



System Number of events: 9,130 (!) New events available

Level	Date and Time	Source	Event ID	Task Category
Information	16-Jan-23 00:01:00	Hyper-V-Hypervisor	1	None

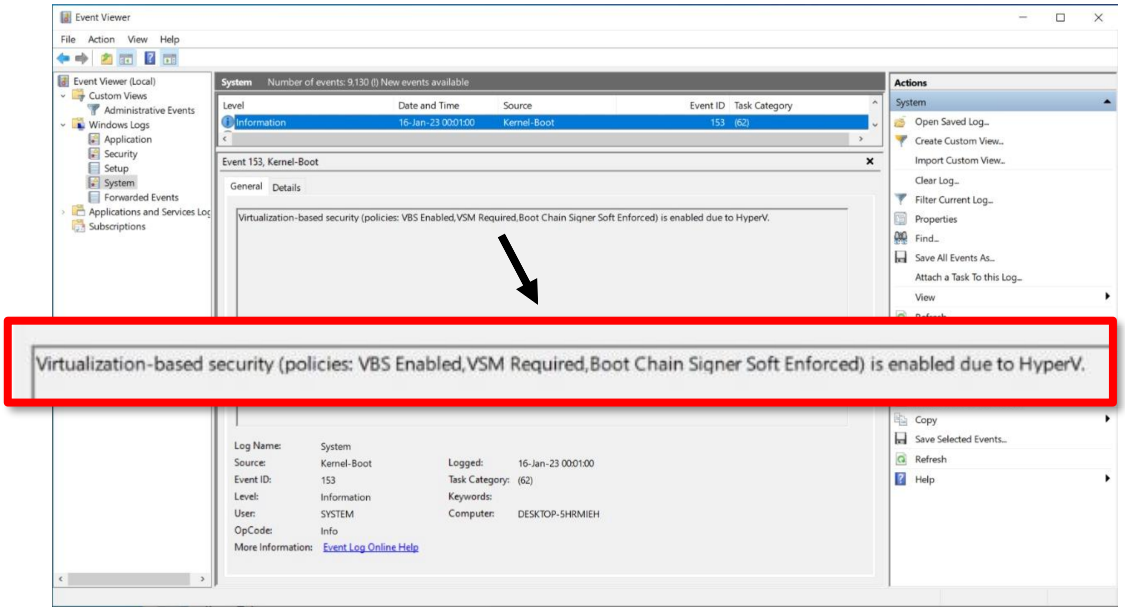
Event 1, Hyper-V-Hypervisor

Hypervisor successfully started.

Log Name:	System		
Source:	Hyper-V-Hypervisor	Logged:	16-Jan-23 00:01:00
Event ID:	1	Task Category:	None
Level:	Information	Keywords:	(70368744177664)
User:	SYSTEM	Computer:	DESKTOP-5HRMIEH
OpCode:	Info		
More Information:	Event Log Online Help		

Hypervisor successfully started.

Fig 6. Hypervisor is successfully started on device

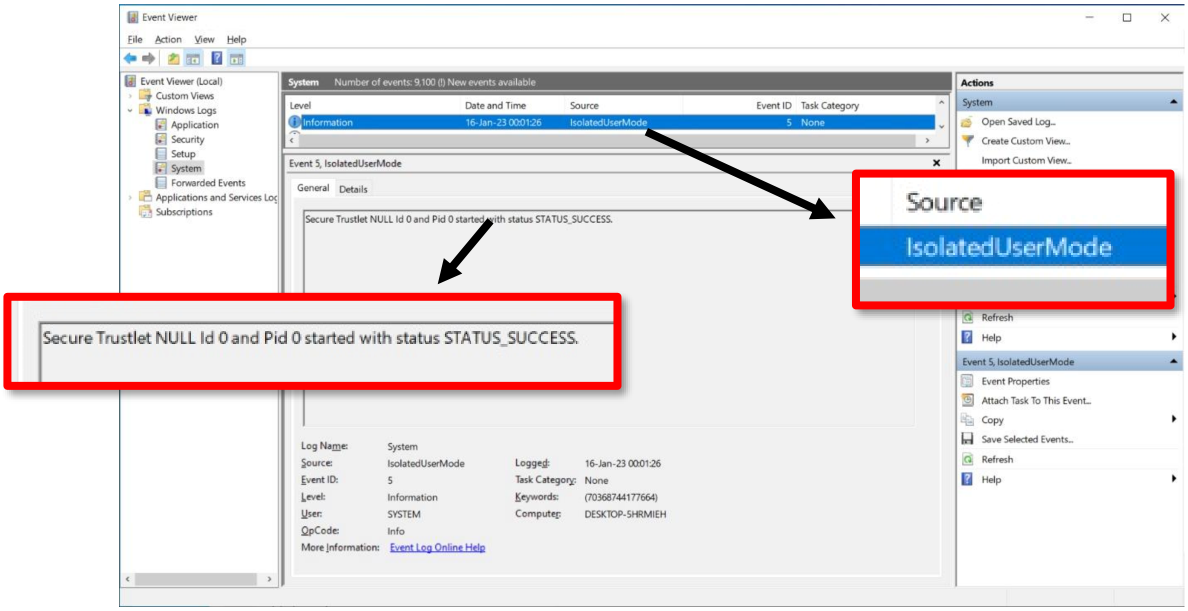


Information 16-Jan-23 00:01:00 Kernel-Boot 153 (62)

Virtualization-based security (policies: VBS Enabled,VSM Required,Boot Chain Signer Soft Enforced) is enabled due to HyperV.

Log Name: System
Source: Kernel-Boot
Logged: 16-Jan-23 00:01:00
Event ID: 153
Task Category: (62)
Level: Information
Keywords:
User: SYSTEM
Computer: DESKTOP-5HRMIEH
OpCode: Info
More Information: Event Log Online Help

Fig 7. VSM is enabled after hypervisor successfully started

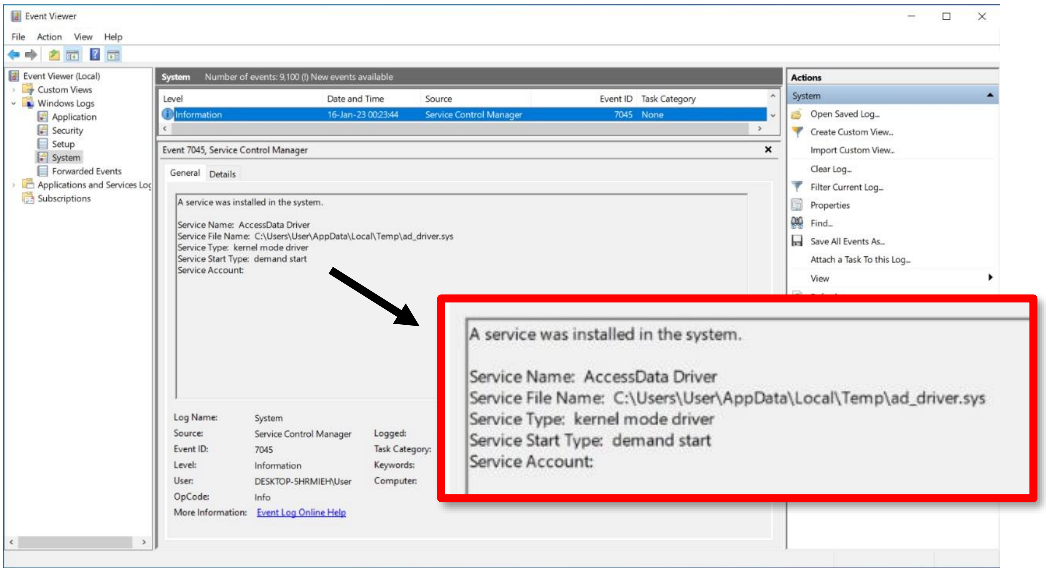


Information 16-Jan-23 00:01:26 IsolatedUserMode 5 None

Secure Trustlet NULL Id 0 and Pid 0 started with status STATUS_SUCCESS.

Log Name: System
Source: IsolatedUserMode
Logged: 16-Jan-23 00:01:26
Event ID: 5
Task Category: None
Level: Information
Keywords: (70368744177664)
User: SYSTEM
Computer: DESKTOP-5HRMIEH
OpCode: Info
More Information: Event Log Online Help

Fig 8. User successfully entering isolated user mode



System Number of events: 9,100 (!) New events available

Level	Date and Time	Source	Event ID	Task Category
Information	16-Jan-23 00:23:44	Service Control Manager	7045	None

Event 7045, Service Control Manager

```
A service was installed in the system.

Service Name:  AccessData Driver
Service File Name:  C:\Users\User\AppData\Local\Temp\ad_driver.sys
Service Type:  kernel mode driver
Service Start Type:  demand start
Service Account:
```

```
Log Name:          System
Source:            Service Control Manager     Logged:
Event ID:          7045                        Task Category:
Level:             Information                 Keywords:
User:              DESKTOP-5HRMIEH\User        Computer:
OpCode:            Info
More Information:  Event Log Online Help
```

A service was installed in the system.

Service Name: AccessData Driver
Service File Name: C:\Users\User\AppData\Local\Temp\ad_driver.sys
Service Type: kernel mode driver
Service Start Type: demand start
Service Account:

Fig 9. AccessData FTK Imager is successfully installed and running in secure mode system

These figs implies that the FTK Imager application is running smoothly without any problem found in the operating system. Different between this study, the latest research by Niken [6], it shows that this application cannot run properly in Windows 10 while VSM environment being turned on.

The analysis in this study, shows that there are several differences from the device processor and operating system being used in this study with the latest study. The differences are shown in Table 4.

TABLE 4
DIFFERENCES BETWEEN THIS STUDY AND THE LATEST RESEARCH

Name	Specification	
	This Study	Latest Research
Device	HP Elitebook 820 G4	Unspecified
Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz (4 CPUs), ~ 2.7GHz	Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, ~ 2.30GHz
Memory	8192MB RAM	20,0GB RAM
BIOS	P78 Ver. 01.16	Unspecified
Operating System	Windows 10 Pro 64-bit (10.0, Build 19044)	Windows 10 Enterprise, Version 21H1 (Build 19043.1526)

This shows some significant differences on the processor and the operating system being used. It is expected that the difference of Windows 10 edition is the one that caused FTK Imager run differently.

The main difference is on the license gave by Microsoft. The enterprise edition is the one with additional license from the Pro edition. The other thing that has be to consider is the security features provide in those different editions of Windows 10. The Pro edition is not providing credential guard and device guard in it. Dissimilar with Windows 10 Enterprise, it is providing credential guard and device guard, which are the main key of the hypervisor that runs VSM [16], [17]. These are the causes that affect the running process of FTK Imager. The Enterprise edition of Windows 10 has more security guard on its operating system that makes the memory acquisition process cannot run properly.

V. CONCLUSION

From the experiment, secure mode environment is not fully support memory acquisition process. The experiment shown that Autopsy version 4.7 is terminated immediately without even have the chance to be opened. Static code analysis shows that is occur caused by a function called __report_gsfailure. This function contains errors in MOV and LEA instructions that takes the action on loading the data to the rax register, which being used to return values from the function. It leads the process to terminate and being aborted by the operating system because it runs on different kernel from the main one, in this experiment is on the safe kernel that created by VSM being turned on.

Other than Autopsy, the experiment on FTK Imager shows that the tools running smoothly for both versions. It leads to comparing this experiment with research by Niken [6], because the result on FTK Imager is different with this study. System analysis method is being applied on FTK Imager and shows that the difference between the edition of Windows 10 can cause different occurrence in the safe mode. It is being found that the Enterprise edition has credential guard and device guard inside the security features, but not in the Pro edition. These security features are defending the operating system from interruption attempt with the help of VSM features on Windows 10.

The result of this whole study shows that VSM environment runs differently while performing memory acquisition process. After doing evaluation for each tool that being tested, it indicates that secure mode on Windows 10 has different library system that cause performance differences in memory acquisition tools. The secure mode itself has different security feature for various editions of Windows 10, specifically in credential guard and device guard features that protect operating system from attacks with the help of Isolated User Mode (IUM). This shows that different edition of Windows 10 can affect the results of the experiment, caused by the differences in its security features.

There are more possibilities that can come in further experiment with secure mode environment. This study is focused on using the VSM features provided by Windows 10. For further research, other memory acquisition tools can be examined with the same condition on Windows 10 or use another operating system. Other Windows version can be the focus for further research on the attempt of acquiring memory for digital forensics act.

VI. REFERENCES

[1] M. A. Hamzah, N. D. Cahyani, and E. M. Jadied, "ANALISIS DAN VISUALISASI TEKNIK PENGHAPUSAN DATA PADA MEDIA PENYIMPANAN YANG MEMENUHI KAIDAH FORENSIK DIGITAL," 2019.

[2] A. Milenkoski and D. Phillips, "Virtual Secure Mode: Architecture Overview.," *hal-03117358*, 2019, [Online]. Available: https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-vbs

[3] K. M. Stewart, "What is Windows 10 isolated user mode (IUM)," *TechTarget*, Oct. 31, 2017. https://www.techtarget.com/searchenterprisedesktop/definition/Windows-10-Isolated-User-Mode-IUM (accessed Apr. 29, 2022).

[4] M. Sikorski and A. Honig, *PRACTICAL MALWARE ANALYSIS*. San Francisco, CA: William Pollock, 2012.

[5] A. Gillis, “Static analysis (static code analysis),” Jul. 31, 2020. https://www.techtarget.com/whatis/definition/static-analysis-static-code-analysis (accessed Feb. 08, 2023).

[6] N. Dwi, W. Cahyani, E. M. Jadied, E. Ariyanto, N. Hidayah, and A. Rahman, “The Influence of Virtual Secure Mode (VSM) on Memory Acquisition,” 2022. [Online]. Available: www.ijacsa.thesai.org

[7] H. K. Brendmo, “Live Forensics on the Windows 10 secure kernel,” Jun. 2017.

[8] H. A. Nugroho and Y. Prayudi, “PENGGUNAAN TEKNIK REVERSE ENGINEERING PADA MALWARE ANALYSIS UNTUK IDENTIFIKASI SERANGAN MALWARE,” 2014, [Online]. Available: www.thehackernews.com

[9] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, “PiOS: Detecting Privacy Leaks in iOS Applications Institute Eurecom, Sophia Antipolis,” Feb. 2011.

[10] M. G. Rekoff and S. Member, “On Reverse Engineering.”

[11] P. Forbrig *et al.*, *Combining Static and Dynamic Analysis for the Reverse Engineering of Web Applications*. 2013.

[12] P. Muntean, M. Fischer, G. Tan, Z. Lin, J. Grossklags, and C. Eckert, “τCFI: Type-assisted control flow integrity for x86-64 binaries,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11050 LNCS, pp. 423–444. doi: 10.1007/978-3-030-00470-5_20.

[13] The Python Software Foundation, “msvcrt — Useful routines from the MS VC++ runtime,” *Python documentation*, Jan. 15, 2023. https://docs.python.org/3/library/msvcrt.html (accessed Jan. 16, 2023).

[14] T. Ahmed and S. Xu, “Shellcoding: Hunting for Kernel32 Base Address,” *IEEE*, 2022, Accessed: Jan. 16, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9798057/authors#authors

[15] D. Hintea, R. Bird, and M. Green, “An Investigation into the Forensic Implications of the Windows 10 Operating System: Recoverable Artefacts and Significant Changes from Windows 8.1,” 2017.

[16] Microsoft, “Compare windows 10 editions for business: Microsoft,” *Microsoft Windows for Business*, 2023, Accessed: Jan. 16, 2023. [Online]. Available: https://www.microsoft.com/en-us/windowsforbusiness/compare

[17] Daniel Todd, “Windows 10 Pro vs Home vs Enterprise: Which is best for your business?,” May 2022, Accessed: Jan. 16, 2023. [Online]. Available: https://www.itpro.co.uk/software/operating-systems/367779/windows-10-pro-vs-home-vs-enterprise-best-for-business