UNIVERSITI PUTRA MALAYSIA

OPERATING SYSTEM KERNEL MALWARE CHARACTERIZATION USING DATA-CENTRIC APPROACH

HARMI ARMIRA BT. MOHAMAD HAR

FSKTM 2018 29

OPERATING SYSTEM KERNEL MALWARE CHARACTERIZATION USING DATA-CENTRIC APPROACH

By

HARMI ARMIRA BT. MOHAMAD HAR

Thesis submitted to the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, in fulfillment of the requirements for the Master of Information Security

JANUARY 2018

COPYRIGHT PAGE

DEDICATIONS

"This sweet dedication goes to respected lecturers, thoughtful friends and supportive family"

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfillment of the requirement for the degree of Master of Information Security

OPERATING SYSTEM KERNEL MALWARE CHARACTERIZATION USING DATA-CENTRIC APPROACH

By

HARMI ARMIRA BINTI MOHAMAD HAR

JANUARY 2018

Supervisor: Dr. Mohd Yunus Sharum
Faculty: Faculty of Computer Science and Information Technology

Malicious software or malware is any malicious code in software that can be used to compromise computer operations, gather sensitive information, gain access to private computer resources and do any illegitimate action on data, host or networks. In this modern technology, malware is rapidly evolved through various stealth techniques to avoid detection. Malware is able to infect and exploit resource from various system platforms. Those evolvements and advanced trick caused code-centric approach becomes less-effective. Especially when the code-centric approach is used to detect OS kernel malware, the approach becomes inflexible as they are good in hiding themselves and cover up their track. Moreover, OS kernel malware also is able to circumvent detection by varying the pattern of code execution. Therefore, this project is proposing a quite brand new approach which is data-centric approach by characterizing the OS kernel malware. This approach tries to detect OS rootkits based on trace pattern found in memory dump content. In order to implement this approach, a Data-Centric OS Kernel Malware Characterization framework is being used. This framework consists of two main components. The first component in this framework is a Dataset of Rootkits Characterization that will create dataset by identifying memory dump content that indicates the trace of rootkits. The second

component which is Determine the Rootkits Presence that able to detect rootkits based on signature created on component one. By collecting the benign and malicious sample, an analysis is being done to create the rootkits signature. This approach is able to detect and calculate the percentage of unknown samples. As for future enhancement, it is better to use more benign and malicious sample to be analyzed. This will increase the accuracy of the result and get more valid rootkits signature.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia Sebagai memenuhi keperluan untuk Ijazah Sarjana Keselamatan Maklumat

PENCIRIAN PERISIAN HASAD TERAS SISTEM PENGOPERASIAN MENGUNAKAN PENDEKATAN BERTERASKAN DATA

Oleh

HARMI ARMIRA BINTI MOHAMAD HAR

JANUARY 2018

Penyelia: Dr. Mohd Yunus Sharum

Fakulti: Fakulti Sains Komputer dan Teknologi Maklumat

Perisian berniat jahat ataupun dikenali sebagai “malware” adalah kod berniat jahat dalam perisian yang digunakan untuk mengkompromikan operasi komputer, mengumpulkan informasi sensitif, mendapatkan akses ke sumber komputer peribadi dan melakukan tindakan tidak sah pada data, tuan rumah atau rangkaian. Dalam teknologi moden ini, “malware” berkembang pesat melalui pelbagai teknik bagi mengelakkan dikesan. “Malware” boleh menjangkiti dan mengeksploitasi sumber dari pelbagai platform sistem. Perkembangan dan kemajuan strategi yang digunakan oleh “malware” menyebabkan pendekatan kod-sentrik menjadi kurang berkesan. Terutamanya apabila pendekatan kod-centric digunakan untuk mengesan “malware” yang mengjangkiti teras sistem pengoperasian. Pendekatan itu menjadi tidak fleksibel disebabkan kelicikan “malware” dalam menyembunyikan diri dan mengaburi kesan mereka. Selain itu, “malware” jenis sistem pengoperasian ini juga dapat mengelak dari dikesan dengan mengubah corak pelaksanaan kod. Oleh itu, projek ini mencadangkan pendekatan yang agak baru iaitu pendekatan data-sentrik dengan mencirikan “malware” jenis teras sistem pengoperasian ini. Pendekatan ini akan mengesan “malware” jenis teras sistem pengoperasian ini berdasarkan pola

jejak yang terdapat dalam kandungan memori terbuang (memory dump). Bagi melaksanakan pendekatan ini, kerangka Penciptaan "Malware OS Data-Centric" digunakan. Rangka kerja ini terdiri daripada dua komponen utama. Komponen pertama dalam rangka kerja ini adalah "Dataset of Rootkits Characterization" yang akan menjana set data dengan mengenal pasti kandungan "memory dump" bagi mendedahkan kesan atau aktiviti "malware" jenis teras sistem pengoperasian . Komponen kedua yang menentukan kehadiran "malware" jenis teras sistem pengoperasian ialah berdasarkan tandatangan yang dibuat pada komponen pertama. Dengan mengumpul sampel yang bersih (benign) dan yang dijangkiti (infected), analisis akan dilakukan untuk mencipta tandatangan"malware" jenis teras sistem pengoperasian. Pendekatan ini dapat mengesan dan mengira peratusan sampel yang tidak diketahui. Bagi peningkatan masa depan, adalah lebih baik menggunakan lebih banyak sampel untuk dianalisis. Ini akan meningkatkan ketepatan keputusan dan mendapatkan tandatangan "malware" jenis teras sistem pengoperasian yang lebih sah.

ACKNOWLEDGEMENT

I feel grateful and thanks Allah SWT because of His bless and mercy. During this period, I am able to learn so many new things especially related to security in Information Technology. First of all, I would like to dedicate this appreciation to my parent Encik Mohamad Har B. Md Ali and Puan Hamlah Haron, who always gives me endless moral support and encouragement. Always give the advice to keep me strong. Next, my appreciation goes to my supervisor, Dr. Mohd Yunus Sharum who always patient in helping me and giving guidelines for improvements. I am very thankful towards his effort and sharing of knowledge to support me until the end of this project. Not to be forgotten, thanks to my fellow friends who are not tired of giving opinion and constructive comments that able to improve my project. May Allah bless all of you.

APPROVAL FORM

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfillment of the requirement for the degree of Master Information Security. The members of the Supervisory Committee were as follows:

DR. MOHD YUNUS SHARUM

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

(Supervisor)

Date: January 2018

DECLARATION FORM

Declaration by graduate student

I hereby confirm that:

- This thesis is my original work
- Quotations, illustrations and citations have been duly referenced
- This thesis has not been submitted previously or concurrently for any other degree at any other institutions
- Intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia (UPM)
- Written permission must be obtained from supervisor and Deputy Vice-Chancellor (Research and Innovation) before thesis is published in book form
- There is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity was upheld as according to Rule 59 in Rules 2003 (Revision 2013-2014). The thesis has undergone plagiarism detection software

Signature: _____ Date: _____

Name and Matric No.: HARMI ARMIRA BT. MOHAMAD HAR (GS 47461)

TABLE OF CONTENTS

Copyright page ..... **i**
Dedications ..... **ii**
Abstract ..... **iii**
Abstrak ..... **v**
Acknowledgement ..... **vii**
Approval form ..... **viii**
Declaration form ..... **ix**
Table of content ..... **x**
List of Table ..... **xi**
List of Figure ..... **xii**

CHAPTER

1 INTRODUCTION **1**
1.1 Background Research 1
1.2 Problem Statement 8
1.3 Research Objective 9
1.4 Research Scope 10
1.5 Research Schedule 11
1.6 Thesis Structure 11
2 LITERATURE REVIEW **14**
2.1 Kernel Malware: Rootkits are stealth malware 14
2.2 Approaches used to Detect OS Kernel Malware Attack 15
2.3 Technique to Trace Rootkits Footprints 16
2.4 Malware Detection Based on Memory Analysis approach 18
2.5 Literature Review Conclusion 20
3 RESEARCH METHODOLOGY **23**
3.1 Project Methodology 23
3.2 Framework 26
3.3 Hardware and Software Requirements 29
4 IMPLEMENTATION **31**

4.1 Implementation of the Framework 31
4.2 Overall Framework Process Flow 39
5 RESULT AND DISCUSSION 41
5.1 Component 1: Dataset of Rootkits Characterization 41
5.2 Component 2: Determine the Rootkits Presence 53
6 CONCLUSION 58
6.0 Conclusion 58
6.1 Future Enhancement 60

REFERENCES 61

LIST OF TABLE

Table 1: Literature Review Summary ……..……..……...21
Table 2: Hardware & Software Requirement …….....30
Table 3: Signature for each sample…….……..…….....….53
Table 4: Test Result for Unknown Sample ……..……...54

LIST OF FIGURE

Figure 1: Malware Detection System……...… 4
Figure 2: Malware Analysis Method…… 5
Figure 3: Waterfall Model…… 22
Figure 4: Data-Centric OS Kernel Malware Characterization Framework…….. 27
Figure 5: Weightage Average Formula…… 36
Figure 6: Data Aggregator Home Interface…….. 37
Figure 7: Data Aggregator DBP Setting Interface…… 35
Figure 8: Data Aggregator Data Analyzer Interface…… 38
Figure 9: Flow Chart of Component 1…….. 39
Figure 10: Flow Chart of Component 2…… 40
Figure 11: Snapshot of additional instruction of Malicious Sample 2 (MS2)…….. 42
Figure 12: Snapshot of additional instruction of Malicious Sample 3 (MS3)…….. 43
Figure 13: Snapshot of additional instruction of Malicious Sample 4 (MS4) ……. 44
Figure 14: Snapshot of hidden service of Malicious Sample 2 (MS2)……. 45
Figure 15: Snapshot of additional service of Malicious Sample 3 & 4 (MS3, MS4)... 45
Figure 16: Snapshot of changing service state of Malicious Sample 1 & 2……. 46
Figure 17: Snapshot of changing service state of Malicious Sample 3 & 4……. 46
Figure 18: Snapshot of suspicious binary path of Malicious Sample 2 (MS2) …….. 47
Figure 19: Snapshot of hidden binary path of Malicious Sample 3 (MS3)……. 47
Figure 20: Snapshot of suspicious process of Malicious Sample 4 (MS4)…….… 48
Figure 21: Snapshot of suspicious driver of Malicious Sample 1 (MS1) …… 49
Figure 22: Snapshot of suspicious driver of Malicious Sample 2 (MS2)……. 50
Figure 23: Snapshot of suspicious driver of Malicious Sample 4 (MS4) …… 50
Figure 24: Snapshot of Unknown module of Malicious Sample 2 (MS2)…….. 51
Figure 25: Snapshot of Unknown module of Malicious Sample 3 (MS3)…….. 52
Figure 26a: Unknown Sample 1 being analyzed…….. 55
Figure 26b: Alert message with percentage of rootkits presence…… 55
Figure 27a: Unknown Sample 2 being analyzed…….. 56
Figure 27b: Alert message with percentage of rootkits presence…… 56

CHAPTER 1

INTRODUCTION

1.0 Introduction

As for the introduction, this chapter will briefly explained the research background, highlight the problem statement, research objective, research scope, research schedule and a brief description of thesis structure.

1.1 Background Research

In this background research, there is a brief description about the related field of study for this project. Background research is mainly to collect information to have understanding in-depth about the related subject. For this project, understanding about malware especially rootkits nature is a foundation to have the overall overview about this project. Besides, study about malware detection system is also important in order to know how malware analysis is being done. Lastly, as this project is using memory analysis to gather information, knowing what is memory analysis and how can it be used in this project is very helpful.

1.1.1 Malware

Malicious software or malware is any malicious code in software that can be used to compromise computer operations, gather sensitive information, gain access to private computer resources and do any illegitimate action on data, host or networks. Malware is able to infect and exploit resource from various system

platforms. There are various classes of malware such as virus, worms, Trojans, bots, backdoors, rootkits and etc. Malware is considered as dangerous as it has the ability to attack main security goals which are confidentiality, integrity and availability.

In this modern technology, malware also rapidly evolve through various stealth techniques to avoid detection. By only depends on the signatures and anomaly-based techniques are not really reliable. Therefore, as a researcher, we need to focus more on finding the generalized and scalable features of malware. Nowadays, malware creator also works on anti-antivirus techniques to give a complex challenge for anti-malware researcher to detect malware. This is because most of anti-antivirus is aims to bypass existing antivirus system. Those are few example of methodologies use by malware creators to avoid anti-virus detection [David et al, 2012].

a) **Code Obfuscation**: Malware code try to look tangled and causing the signature-based approach failed by includes some unnecessary jumps, replacing unused registers, no-op instructions and others.
b) **Encryption**: Encrypted malware consists of encrypted part that able to beat easily signature-based approach
c) **Polymorphism & Metamorphism**: Polymorphism makes use of payload while metamorphism is able to change itself or do self-mutating. These two methodologies are powerful and difficult to be detected.

1.1.2 Operating System Malware (Rootkits)

Rootkit is one kind of malware that normally related to OS. With a term that made up of two words which 'root' is commonly giving user full administrative over the system, while 'kit' is set of application that carries out task or administrative process [Hili et al, 2010]. There is various type of rootkit available and type rootkits suit are depends on the specific system of the OS. Rootkits are considered as one of stealth malware as it is very difficult to be detected. Moreover, it is able to modify and manipulate OS modules once infected and is also able to cover its tracks. Some rootkits are able to attack kernel level which gives them complete control over the OS.

There are common approaches used by the attacker to distribute rootkits. One of them is by infect existing and legitimate web server that cause user unaware of the risk they confront. Another approach is where the attacker creates a common application or common files which appear as legitimate yet contain harmful script once executed. The reasons why rootkits are difficult and complex to be detected are the attack methods evolve along the time. Rootkits are able to alter the output of common system administrative command such as a list of running process and all the opened port [Prakash et al, 2013]. They will give an inaccurate report that indicates everything is clean. Even though there are several methods that helps in detecting rootkits, but still it may lead to false results. The main things about rootkits are they able to modify software application that may lead to report fake results. In addition, rootkits are designed to be remaining undetected, so they are expert on how to cover their track and hiding themselves. Among all the common method on detecting

rootkits, behavioral analysis is more reliable. However, it is time-consuming and needs specific knowledge as it needs some analyze process.

Rootkits can be divided into two types generally which are application rootkit and kernel rootkit. Application rootkits is a rootkit that establish at the application layer, while kernel rootkits is a rootkits that is able to infect deep into kernel layer. This project focuses more on kernel rootkits which are more powerful as they are difficult to be detected. Normally kernel rootkits are able to hide a process and files, hiding network connection, able to redirect file execution and etc. However, the activity or the behavior of rootkit leaves a footprint that later can be used as a pattern to detect their presence.

1.1.3 Malware Detection System

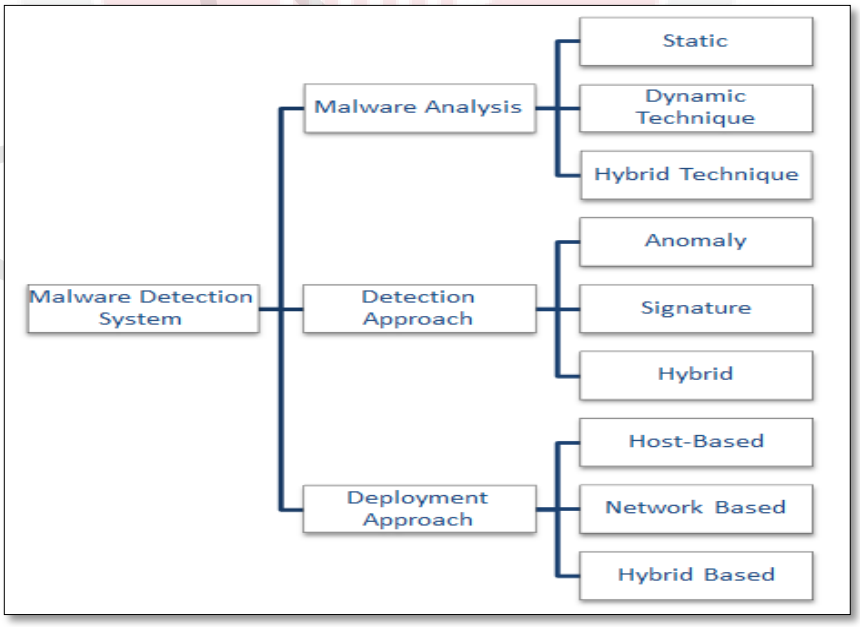

Figure 1: Malware Detection System

Malware detection technique needs to be periodically updated and must always one step further than those entire anti-antimalware products. As shown in Figure 1, Malware Detection System consists of three main parts which are analysis technique or as known as malware analysis, detection approach and deployment approach [Faizal et al, 2016]. Malware analysis is the important part need to be considered in order to achieve an effective technique and approach. Malware analysis is a process to perform analysis and study the components of malware's code and identify the characteristic of their behavior. Besides, as shown in Figure 2, in malware analysis there are three main techniques can be used which is static technique, dynamic technique and hybrid. The static technique is being done without running the malware while dynamic technique will execute malware. Hybrid is the combination of Static and dynamic technique [Sie, 2015].

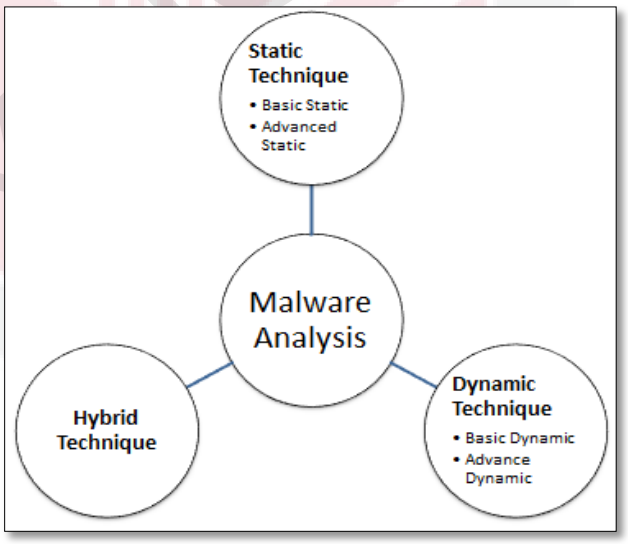

Figure 2: Malware Analysis Method

This project will use a dynamic technique. There is two type of dynamic technique which is basic dynamic analysis and advanced dynamic analysis. Basic dynamic will use a virtual machine to do malware analysis and monitor the process and the behavior of malware, while the advanced dynamic will further analyze in depth about the malware. The second part of malware detection system is detection approach. Detection approach can be used are anomaly, signature or hybrid.

As for deployment approach, there are host-based, network-based or hybrid based. This project is to implement advanced dynamic analysis as it will monitor the behavior of the malware and do some further analysis. The analysis in this project is being done towards memory dump. Further detail on the process flow and other important processes will be explained in Chapter 3. In addition, these projects also use the combination of anomaly and signature for detection approach and use host-based for deployment. By monitor the trace of anomaly behavior of the rootkits in the memory dump, a signature is created in order to use as detection approach.

1.1.4 Rootkit Detection based on Volatility Information in RAM

Memory analysis is using memory image (memory dump) to determine about the running program, operating system information and overall state of the computer [memory forensic rootkit]. A raw memory dump is a complete snapshot of memory that records content of system memory, data from processes that were running when the memory dump was collected.

Normally memory dump is used as evidence in court as it consist volatile information [Amari, 2009]. The memory image can be used to determine information about running programs, operating system states and also to locate deleted or temporary information as long as the machine is on. This because memory dump consist of volatile data which be recorded when the machine is on and will be lost when the machine is off. There are several forensics tools that are able to acquire memory dump such as FTK Imager. In this project, FTK imager is used to acquire the memory dump from Windows 7. Moreover, there are several software and tools that can be used to analyze the sample of memory dump. One of them is a tool in Kali Linux (Volatility Framework).

As mention previously, memory dump are normally used as evidence as it may consist valuable information that may help in investigation. However, there is a plugin in Volatility Framework that can be used to trace the footprint of rootkits [Halleigh et al, 2014]. There are supported plugin for MS Windows, Linux and Mac OS memory dump. They are many of available plugins such as process memory, kernel memory and objects, networking, registry, malware and file system. Thus, we need to identify which plugin is suitable to be used and are able to show a significant result to trace rootkits activities and behavior. The detail on chosen plugin will be explained in Chapter 3.

1.2 Problem Statement

Modern malware tends to become tricky and confusing the malware scanner as they able to combine several characteristics of the undesirable program from different classes [Hili et al, 2010]. Besides, they are also evolved rapidly in every aspect especially on advancing their attack strategies [Rhee et al, 2014]. To avoid detection, there is some malware that use obfuscating techniques such as reuse a legal code, capable to modify their structure (polymorphism) and also able to replace some routine of the targeted resource (stealth virus) [Ford & Howard, 2007]. Those evolvements and advanced trick cause code-centric approach become ineffective. Especially on OS environment, it is very difficult to detect malware such as rootkit as they able to modify and replaced OS module to cover their track [Guri & Poliak, 2015].

Code-centric becomes more unreliable to deploy on detection of OS kernel malware as they good in hiding themselves [Musavi & Kharrazi, 2014]. Moreover, not only able to trick code-centric approach, OS kernel malware also able to circumvent detection by varying the pattern code execution that will confuse behavior-based malware detectors [Sharif et al, 2008]. This shows that several approaches like code-centric and malware-based behavior tend to become unreliable with the evolvement of malware.

The main problems that need to be explored are OS malware which is normally known as rootkits are they are expert in hiding their presence and able to subvert

normal operating system behavior. There are several techniques used by rootkits to subvert OS behavior such as hooking OS APIs and system call table (SDT), hiding in unused space on machine's hard disk and infecting the master boot record (MBR) [Respon & 2010]. SDT contains a data structure that is able to process system call. By manipulating this table, rootkits are able to insert malicious instructions. Kernel rootkits are not only able to add new code but also have the capability to delete and replace operating system code. This capability makes kernel rootkits becomes inevitable once infect the operating system. Therefore by only depends on code-centric approach, it could be impossible to detect kernel rootkits. Besides, a code-centric approach normally used specific characteristics for each rootkit. This makes it less flexible when it comes to unknown or new rootkits attack.

1.3 Research Objective

The objective of this project generally is to develop a model based on a data-centric approach that is able to detect operating system rootkits based on trace pattern found in the memory dump. The objectives are specifically defined as below;

1.3.1 Create a standard and general signature of operating system rootkits based on characteristic and pattern found in the memory dump.

As mention previously in problem statement section, there are a lot of previous work that proposes on OS rootkit signature based on either code-centric or data-centric approach. However, most of researcher proposed a signature that mainly for specific rootkits or even for specific behavior [Lin et al, 2011][Davide et al,

2010][Case & Iii, 2015][Case & Richard, 2016][Korkin & Nesterov, 2014]. This specific rootkits signature are vulnerable against evolve or advanced rootkits attack as they create a fix signature. With the evolution of rootkits structure, attack strategy and behavior, the fixed and specific signature becomes less flexible to detect their presence. Thus, to encounter this problem, the first objective is proposed.

3.1.2 Able to detect operating system rootkits by using signature created.

The second objective aims to use the created signature in first objective to detect rootkits with variance type. As this project proposes a standard and general signature of rootkits, we aim to detect rootkits presence without depends on specific rootkits structure, attack strategy and behavior. The general rootkits signature is proposed based on the characteristics and pattern found in the memory dump content. The characteristics and pattern are collected from various types of rootkits. This is to standardize the signature and is applicable towards various rootkits.

1.4 Research Scope

As for this project, there is two operating system involve which is Windows 7 (SP0). Windows 7 is being chosen as operating system as a medium to collect memory dump sample. There are two types of memory dump that will be collected from Windows 7 which is a benign sample (clean) and malicious sample (infected).

Besides, we are used Kali Linux is to provide analysis tool which is Volatility Framework. Memory Dump that is collected by using FTK Imager (forensic tool) will be analyzed in Volatility Framework. VMware Workstation is a virtual boxes

that being used to create new virtual machine where Windows 7 (SP0) and Kali Linux are installed. There are also existing text editor tool which is DiffMerger that are being used to perform static analysis on the output of memory dump.

In addition, in this project there 6 selected rootkits being used. Those rootkits are being executed in Windows 7 (SP0). In addition, in this project, it is must to know on how to implement data-centric approach in order to generate malware signatures based on the characteristics and pattern found in memory dump content. This project also covers on static data analysis where further analysis of data obtains in order to get a trace pattern of OS kernel malware behavior. As for the malware detection system, this project using advanced dynamic technique for overall malware analysis. This project involved further analyze against the rootkits behavior's pattern in memory dump content. This project used hybrid detection approach where an anomaly behavior is being monitored and analyzed to create a signature. Lastly, project only covers on host-based for deployment approach and not for either network based or hybrid based.

1.5 Research Schedule

This project is one year period which starts in January 2017 and expected to finish in January 2018. Generally, this project has six activities which are project implementation plan, knowledge gathering, experimentation design, implementation and development, testing and evaluation and lastly report write up. Each project activities have their own milestones that need to be achieved. The details for project activities milestones are explained in Chapter 3.

1.6 Thesis Structure

The structure of this thesis consists of six chapters including Introduction, Literature Review, Methodology, Project Implementation, Result and Discussion and last chapter is Conclusion.

Chapter 1 is briefly explaining the introduction of background study of the related subject towards this project. Besides, this chapter also consist problem statement, research objective, research scope, research result expectation and thesis structure. Research objectives are derived from problem statement and expected to be achieved at the end of this project. Research scope and research schedule are to highlight the scope of this project and to ensure this project is on the right track according to schedule stated.

Chapter 2 is a list of literature review for this project. A literature review is a source of research article and journal that being used to give more understanding about related topic. This chapter is important as it is to ensure this project is possible to be done and to avoid any duplication of previous work (research gap). Besides, this chapter act as knowledge resource that helps to do improvement, tips, proof of concept based on previous research. This is able to help to increase the success rate for this project.

Chapter 3 is the chapter that explained methodology that being used to develop this project. A methodology is one of essential elements in every project as it to

ensure the project is properly plans and can be executed smoothly. In this chapter also explained the framework that being used in this project.

Chapter 4 is project implementation and development. In this chapter, the approach used is being explained in detail. Besides, the design and the functionalities of the approach also are highlighted. This chapter provides overall and process flow chart for this project. All the detail according to the implementation of the approach also can be found in this chapter.

Chapter 5 is a explaining the result. In this chapter, all the result and finding related to this project will be provided here. An evaluation of the result and the discussion are explained in this chapter.

Chapter 6 is the chapter of conclusion for this project. Besides, there is also suggestion for future enhancements that can be done. This chapter also concludes the whole project, result and the achievement while doing this project.

REFERENCES

[1] David, E., Agusti, S.., & Antoni, M. (2012).Computational Intelligence for Privacy and Security (Vol.394). 3642252370, 9783642252372

[2] Hili, G., Mayes, K., & Markantonakis, K. (n.d.). The BIOS and Rootkits, 369–381. https://doi.org/10.1007/978-1-4614-7915-4

[3] Prakash, A., Venkataramani, E., Yin, H., & Lin, Z. (2013). Manipulating Semantic Values in Kernel Data Structures : Attack Assessments and Implications.

[4] Faizal, M., Razak, A., Badrul, N., Salleh, R., & Firdaus, A. (2016). Journal of Network and Computer Applications The rise of "malware" : Bibliometric analysis of malware study. Journal of Network and Computer Applications, 75, 58–76. https://doi.org/10.1016/j.jnca.2016.08.022

[5] S, S. Y. (2015), Implementation of Malware Analysis using Static and Dynamic Analysis Method, 117(6), 11–15.

[6] Amari, K., (2009), Techniques and Tools for Recovering and Analyzing Data from Volatile Memory, SANS Institute

[7] Halleigh, M., Case, A., Levy, J.,& Walters, A.,(2014) The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux and Mac Memory, John Wiley & Sons, Inc

[8] Rhee, J., Riley, R., & Lin, Z. (2014). Data-Centric OS Kernel Malware Characterization, 9(1), 72–87.

[9] Ford, E. R., & Howard, M. (2007), How Not To Be Seen, 67–69.

[10] Guri, M., & Poliak, Y. (2015). JoKER : Trusted Detection of Kernel Rootkits in Android Devices via JTAG Interface. https://doi.org/10.1109/Trustcom.2015.358

[11] Musavi, S. A., & Kharrazi, M. (2014). Back to Static Analysis for Kernel-Level Rootkit Detection, 9(9), 1465–1476.

[12] M.Sharif, A. Lanzi, J. Giffin,. W. Lee, "Impeding Malware Analysis using conditional code obfuscation", in Proc. 15th Annu. NDSS, 2008, pp. 1-30

[13] Response, S. (n.d.). Rootkits What are Rootkits ?, (figure 1), 1–9.

[14] Lin, Z., Rhee, J., Zhang, X., Xu, D., & Jiang, X. (2011.). SigGraph : Brute Force Scanning of Kernel Data Structure Instances Using Graph-based Signatures.

[15] Davide Balzarotti, Marco Cova, Christoph Karlberger, Christopher Kruegel, Engin Kirda, sand Giovanni Vigna. Efficient Detection of Split Personalities in Malware. In Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS'10),(2010).

[16] Monirul Sharif, Andrea Lanzi, Jonathon Giffin, and Wenke Lee. Impeding Malware Analysis Using Conditional Code Obfuscation. In Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08),(2008).

[17] Case, A., & Iii, G. G. R. (2015). Advancing Mac OS X rootkit detection. Digital Investigation, 14, S25–S33. https://doi.org/10.1016/j.diin.2015.05.005

[18] Case, A., & Richard, G. G. (2016). Detecting objective-C malware through memory forensics. Digital Investigation, 18, S3–S10. https://doi.org/10.1016/j.diin.2016.04.017

[19] Ford, E. R. (2007)., How Not to Be Seen II, 65–68.

[20] D.-H. You and B.-N. Noh, "Android platform based linux kernel rootkit," in Malicious and UnwantedSoftware (MALWARE), Fajaro, 2011.

[21] H. S. K. W. Y. J. J. &. J. S. Sun, "TrustDump: Reliable Memory Acquisition on Smartphones," in ESORICS, 2014.

[22] Kane, P. O., Sezer, S., Mclaughlin, K., & Im, E. G. (2013)., SVM Training Phase Reduction Using Dataset Feature Filtering for Malware Detection, 8(3), 500–509.

[23] CARBONE, M., CUI, W., LU, L., LEE, W., PEINADO, M., and JIANG, X. Mapping kernel objects to enable systematic integrity checking. In Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS) (November 2009).

[24] Cui, W., Peinado, M., & Chan, E. (2011.). Tracking Rootkit Footprints with a Practical Memory Analysis System.

[25] Denzel, M., Stüttgen, J., & Stefan, V. (2015). Acquisition and analysis of compromised firmware using memory forensics, 12, 50–60. https://doi.org/10.1016/j.diin.2015.01.010

[26] Spainhower, M. (2008). Feasibility Analysis of DTrace for Rootkit Detection, (March).

[27] Shahzad,F., Farooq, M., (2013). In-Execution dynamic malware analysis and detection by mining information in process control block of Linux OS, Information Sciences, 231, S45-63

[28] Muthumanickam,K., Ilavarasasan,E., (2015). CoPDA: Concealed Process and Service Discovery Algorithm to Reveal Rootkit Footprints . pp 1 - 15, 28(1), 1–15.

[29] Bill Blunden, *The rootkit arsenal*, by Jones & Bartlett Publishers, edition 1, May 4, 2009, pp 138-139.

[30] Romana, S., Jha, A. K., Pareek, H., & Eswari, P. R. L. (2013.). Evaluation of Open Source Anti-Rootkit Tools.

[31] Fu, D., Zhou, S., Cao, C., &, A. R. (2010). A Windows Rootkit Detection Method Based on Cross-View, 1–3.

[32] Voitovych, O., Kupershtein, L., & Pavlenko, I., (2017) Hidden Process Detection for Windows Operating Systems (PICS&T), 461-464

[33] Shahzad, F., Shahzad, M., & Farooq, M. (2013)., In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS. *Information Sciences*, *231*, 45–63.

[34] Xie, X., & Wang, W. (2013). Rootkit Detection on Virtual Machines through Deep Information Extraction at Hypervisor-level, 498–503.

[35] Cao, Y., Miao, Q., Liu, J., & Li, W. (2013). Osiris : A Malware Behavior Capturing System Implemented at Virtual Machine Monitor Layer, *2013*.