Towards Automated Malware Behavioral Analysis and Profiling for Digital Forensic Investigation Purposes

Ahmed F.Shosha, Joshua I. James, Alan Hannaway,
Chen-Ching Liu and Pavel Gladyshev

UCD School of Computer Science and Informatics,
University College Dublin, Dublin, Ireland
{Ahmed.Shosha,Alan.Hannaway}@ucdconnect.ie
{Joshua.James,Liu,Pavel.Gladyshev}@ucd.ie

Abstract. Digital forensic investigators commonly use dynamic malware analysis methods to analyze a suspect executable found during a post-mortem analysis of the victim's computer. Unfortunately, currently proposed dynamic malware analysis methods and sandbox solutions have a number of limitations that may lead the investigators to ambiguous conclusions. In this research, the limitations of the use of current dynamic malware analysis methods in digital forensic investigations are highlighted. In addition, a method to profile dynamic kernel memory to complement currently proposed dynamic profiling techniques is proposed. The proposed method will allow investigators to automate the identification of malicious kernel objects during a post-mortem analysis of the victim's acquired memory. The method is implemented in a prototype malware analysis environment to automate the process of profiling malicious kernel objects and assist malware forensic investigation. Finally, a case study is given to demonstrate the efficacy of the proposed approach.

Keywords: Dynamic Malware Analysis, Kernel Object Profiling, Malware Investigation, Memory Forensics, Post-Mortem Analysis.

1. Introduction

Malware, or malicious software, has become a commonly used tool to commit crimes on the Internet, and poses significant threat to the security of computer systems and privacy of computer users. To defend against malware, a large body of computer security research has resulted in various techniques to analyze, detect and eliminate malware [1-8]. Although proposed approaches assist malware analysts in accomplishing their mission, advanced malware countermeasure techniques have been developed to generate variants of the malicious code in an attempt to elude detection from traditional methods. Further, the substantial increase in discovered malware samples every day negatively impacts the effectiveness of traditional static analysis approaches. As such, highly automated dynamic techniques have been called for [9]. A variety of automated dynamic malware analysis approaches have been

proposed to cope with the large number of discovered malware samples. These dynamic methods were implemented in various sandbox solutions to provide the required process automation, and assist malware analysts in acquiring required knowledge about the malicious code's behavior [10]. A sandbox, in this work, refers to a managed virtual environment with a pre-determined software configuration used to implement proposed methods and to observe a behavior of a malicious binary through its execution process [10].

From a digital forensic investigation perspective, when investigators are confronted with an investigation involving a suspect executable, different incident response procedures are followed to analyze and investigate the suspect binary. Dynamic malware analysis methods proposed in computer security research are commonly used to allow an investigator to understand the behavior of a suspect executable. Analysis of extracted traces, correlating evidence and artifacts to the suspect executables' behavior, however, is manually conducted by the investigator, and solely relies on his or her expertise. This manual process is time consuming, error prone and allows for inconsistent interpretation of malicious evidence which threatens the integrity of the investigation [11]. Moreover, the use of dynamic malware analysis methods for forensic investigation purposes has a number of limitations. Currently proposed methods are designed to assess the behavior of malicious code for signature development purposes, and have not been designed specifically considering the concepts and principles of digital forensic investigations. Thus, employing these methods in malware forensic investigations may result inaccurate conclusion.

This work highlights the limitations of the use of currently proposed dynamic malware analysis methods applied to digital forensic investigations, and proposes a set of improvements to utilize these methods for forensic investigation purposes. To this end, a method for dynamically profiling the kernel memory of malware objects is proposed. The proposed method allows for automated identification and extraction of malicious kernel objects from a victim's acquired forensic memory image during a post-mortem forensic analysis. In addition, it can be extended to profile different behavioral aspects of malware execution, and allow an investigator to automate the process of malware traces detection in a post-mortem forensic analysis of the victim's computer system. To demonstrate the applicability of the proposed method, a prototype forensic-specific dynamic analysis sandbox solution has been developed and implements the proposed profiling technique. Developed sandbox is evaluated through a case study involving profiling a commonly used malware tool-kit that emerged over the last few years to commit financial crimes on the Internet. Developed profiles are then used to automate the analysis of dynamic kernel memory during post-mortem forensic analysis and automatically identify malware related kernel objects.

To summarize, the contribution of this paper is as follows:

- This work highlights the limitations of the use of currently proposed dynamic analysis methods in malware forensic investigations, and outlines required improvements to utilize the capabilities of these methods for digital forensic investigation purposes.
- This work proposes a dynamic profiling method applied to dynamic kernel memory to automate the process of identification and extraction of malicious

kernel objects in acquired forensic memory images during a post-mortem forensic analysis.
- This work present a prototype dynamic malware analysis sandbox for digital forensic investigation purpose based on the proposed dynamic kernel memory profiling method.

Paper Organization. In section 2, limitations of currently proposed dynamic malware analysis methods in forensic investigations are described. In section 3, profiling of dynamic kernel memory for digital forensic investigation purposes is presented and described in details. Section 4 describes the prototype implementation of the proposed approach, and gives a case study. Section 5 gives a discussion about the proposed method and outlines future research work. Finally, section 6 concludes the paper.

2. Limitations of Dynamic Analysis Methods From a Digital Forensic Investigation Perspective

Dynamic analysis of malware is an automated approach to identify a behavior of malicious program through observation of the program's execution in a managed environment [12]. Typically, malicious programs are automatically loaded into a managed virtual machine environment and executed. Interactions between the malicious program and an operating system are observed to provide human analysts an overview about the sample's behavior and whether further analysis is required or not. Observed interactions of the malicious program in monitored operating systems include which system calls are invoked, and arguments used to interact with the operating system kernel. Finally, a detailed report about the program's activities, i.e. file activities, Windows Registry activities, and networking activities, are provided to the analyst. Such information allows a human analyst to identify if a program under analysis is a new malware sample, a variant sample or a benign program. Based on the analyst's decision, proper detection signature is developed. In contrast, a number of anti-analysis techniques have been developed by malware authors to disrupt malware analysis process, and impede further investigations [13, 14].

Although currently proposed dynamic analysis methods substantially automate and improve the process of malware analysis and malicious code signature development in computer security research, the use of these methods is limited in digital forensic investigations. Thus, a part of malware analysis for digital forensic investigations is accomplished manually despite the fact that it can be automated, if digital forensic investigation objectives were initially considered and integrated into the design of these methods. More important, relying on results of currently proposed methods may contribute in resulting inaccurate forensic investigation conclusions.

This section highlights a number of limitations that hinder utilization of currently proposed dynamic malware analysis methods in digital forensic investigations, and proposes a set of improvements that, if considered, assist in automating malware investigation and preserve the integrity forensic analysis.

2.1 Multiple Malicious Execution Paths

Malware developers employ different methods to impede dynamic analysis of malware and malicious code investigation [14]. A prevalent feature in malware is the frequent collection of intelligence about the surrounding environment and attempting to detect whether it is an analysis or debugging environment. If an analysis environment is detected, malware may suppress its execution and terminate malicious payload installation, or may execute a different execution path that results in benign traces in an attempt to evade the human analyst. This behavior is termed "*malware's evasion personalities*" [15]. To defend against evasion personalities, various approaches have been proposed to disguise the analysis environment, so that, it becomes transparent to a malware. Although proposed disguising methods to defend against evasion personalities substantially contribute to the intended analysis goal, a possibility of existence of multiple execution paths is still valid. Malware may have different malicious payloads or have different behaviors based on certain properties of the compromised environment: the existence of a predetermined Internet browser version, or installation of specific software or hardware, for example. Since currently implemented dynamic analysis methods do not consider tracking multiple execution paths [16] and developed sandbox solutions cannot consider all possible environment configurations, analysis may result in an execution path that has never been executed on the victim system subject of forensics investigation. This incomplete analysis could lead digital forensic investigators to reach an invalid conclusion based on incomplete knowledge of the behavior of the malware.

To overcome multiple execution paths in computer security research, paths tracking techniques have been proposed to execute all possible paths in malicious programs [16]. Unfortunately, proposed techniques are computationally expensive when applied to thousands of malware samples collected every day.

Investigation of all possible malware execution paths can be associated with observations of the state of the system to determine possible explanations that could have resulted in observed system state. Formal theories have been proposed to provide required explanations in the context of multiple execution paths, and to reconstruct events related to a certain execution path [17, 18]. Although these theories have applications in different forensic investigation domains, an application to malware evasion personality detection is still missing. Thus, inclusion of these approaches to malware analysis provides more information to assist investigators in deriving reasonable conclusions.

2.2 Interrelation Between Observed Objects

Dynamic malware analysis methods monitor the interactions between a malware sample and an operating system kernel [19, 20], e.g. invoked system calls and its arguments. Other methods, such as those proposed in [21, 22] not only observe objects interactions, but also, profile the interaction patterns such as evolving pattern of a malicious object's data structure in dynamic kernel memory. Profiled patterns are further used to derive a malware detection signature. In digital forensic investigations of malware, interrelations between observed objects are essential to deduction of further actions invoked by a malicious object [23]. If relations between observed

objects are not properly defined, it may not be possible to infer an instance of an action. That is, investigators are required to manually define the relations between observed objects. Currently proposed dynamic malware analysis methods do not observe and define the mutual relation between malicious objects, although, there are various extensions that can provide necessary information about malicious objects interrelationships [24].

Different methods allow tracking information flow between objects, denoted as dynamic taint analysis, which is considered a complementary approach to dynamic analysis approaches [25]. In dynamic taint tracking, information is labeled and tracked throughout program execution for different purposes. More precisely, propagation of labeled information in dynamic tainting systems is tracked in the context of a malicious objects' execution. Currently proposed dynamic tainting and tracking systems focus on tracking data between objects; however, data propagation paths can be used to derive the interrelations between observed objects. Such derived information allows automation of the process of object interrelation construction, and allows investigators to infer further actions based on defined objects relationships. That is, extending dynamic taint tracking to consider dynamic identification of interrelationships between observed malicious objects and integration of such methods in dynamic analysis approaches is essential for digital forensic investigation, and assists investigators in automating the forensic analysis processes based on object relationships.

2.3 Profiling Dynamic Kernel Objects

Memory forensics is an important portion of digital forensic investigation process when malware is concerned. Various signature-based approaches have been proposed to extract kernel data structures from dynamic kernel memory [26]. These methods scan the dynamic kernel memory to detect and extract different kernel data structure types such as processes, threads, network or VAD objects in Windows operating system kernel [27, 28]. Forensic analysis of extracted objects, and determining if an object belongs to a malware, is a manual process that relies on the investigator's expertise. Forensic analysis of kernel data structure objects requires, as well, deep knowledge of the operating system internals and techniques employed by malware to disrupt investigation through the manipulation of the kernel object characteristics. Moreover, specification of the kernel data structures are likely to change with new builds of the operating system kernel. Thus, manual investigation of the kernel data structure in acquired memory is a significant challenge for forensic investigators.

In computer security research, different approaches have been proposed for automated profiling of kernel objects characteristics in dynamic kernel memory, to assist malware signature development process [22]. Proposed methods are designed to identify the evolving patterns of kernel data structures in memory and profile such pattern. However, these methods are insufficient for forensic analysis of kernel data structure, as they do not allow for automated identification of malicious objects in post-mortem forensic investigations based on developed profiles.

Profiling for digital forensic investigation purposes has been proposed in different investigative domains [29]. However, profiling malware behavior for digital forensic investigation is still missing. A profiling method to automate forensic

identification and extraction of malicious objects will, significantly, assist the process of malware forensic investigation and memory forensic analysis.

3 Profiling Dynamic Kernel Memory

In this section, a method for profiling dynamic kernel memory for digital forensic investigation purposes is presented. The proposed method allows for automated identification of malicious kernel objects in post-mortem forensic analysis of acquired memory.

Dynamic kernel memory is a memory portion where dynamically allocated kernel data structure objects are present. Dynamic kernel memory recently became a target of an increasing amount of kernel level malware such as rootkit attacks [30]. These attacks employ advanced stealth techniques to control and manipulate an operating system kernel. For example, Direct Kernel Object Manipulation (DKOM) attack allows rootkits to hide malicious kernel objects in the operating system kernel through manipulation of malicious kernel object's characteristics [30]. Other attacks such as hijacking kernel execution – denoted as Kernel Object Hooking attack (KOH) [31]– allow kernel level malware to execute compromised code after hijacking the kernel code control flow.

Digital forensic investigators of dynamic memory are required to investigate various kernel level data structure types to identify a presence of rootkits, and existence of malicious kernel objects. As previously discussed, this investigative process has limitations that may compromise the investigation's integrity. Thus, an approach for automated forensic investigation of dynamic kernel objects is required.

In the proposed method, a procedure to monitor kernel object's characteristics is proposed and utilized to develop a profile for malicious kernel objects. Developed profiles will allow investigators to automatically determine kernel objects related to malware in an acquired forensic memory image during post-mortem forensic analysis based on Object-To-Profiles matching procedure.

3.1 Profiling Malicious Kernel Objects for Forensic Investigation Purposes

Program execution process in the operating system requires allocation of memory regions to the program to execute its instructions, and creation of kernel objects in dynamic kernel memory to manage the program execution. These kernel objects control every aspect of the program's execution in the operating system kernel. For example, `EPROCESS` in Windows operating systems [32] or `task_struct` in UNIX based systems represent and manage running program's processes and threads in the operating system kernel.

Since kernel object data structures are formally defined by the operating system code, and instances of these objects are allocated in dynamic kernel memory, investigators attempt to differentiate between benign kernel objects and malicious kernel objects. This process is essential to determine which memory regions are allocated to malware, and which regions are suspicious but non-conclusive and require further analysis.

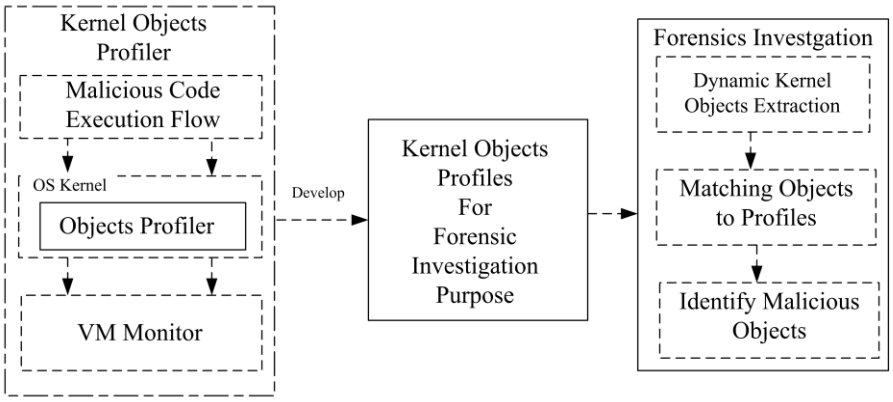

Figure 1: Kernel Object Profiler Process Model

To automate the process of malicious kernel object identification in post-mortem memory analysis, the proposed method profiles the characteristics of a malicious kernel objects in dynamic kernel memory. When a program is being executed in, there exist a unique set of characteristics of kernel object's properties that identifies the program. Determining such characteristics and monitoring its values, while program code is being executed, allows for development of an object profile that can be used to assist kernel objects investigation. To determine characteristics of malicious kernel objects, memory monitoring and introspection technique are employed to observe memory regions allocated to the dynamic kernel objects.

The proposed profiling process model is shown in Figure 1. A malicious code executed instructions are monitored through a dynamic analysis method in a managed virtual environment. Through malicious code execution, snapshots of memory allocated to the kernel objects that represent malware execution is acquired at each executed instruction, and are added to the object profile for further use in a digital forensic investigation. Acquired memory snapshots are automatically matched to the kernel object's definition to identify kernel object property values and determine values that have changed as a result of instructions execution. Finally, at post-mortem memory investigation, developed profiles are used to automate malicious kernel object identification. This is accomplished by extracting dynamic kernel objects from acquired memory and automatically match extracted objects with developed profiles.

3.2 Kernel Object Memory Profiling Formalization

This section presents a formalization used in profiling memory allocated to malicious kernel objects, and determining object properties at different execution states. A malicious kernel object O_m represents a malicious code execution in an operating system kernel. Memory region μ is a dynamic kernel memory space allocated to O_m. A kernel object O_m holds a set of properties ρ_n that used by the operating system kernel to manage the program execution, such that:

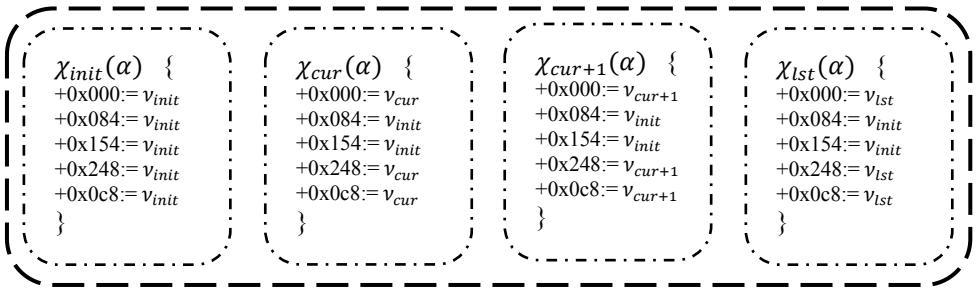

Figure 2: an Example of Kernel Object Profile

$$O_x = \{\rho_1, \ldots \rho_k | \, \rho_j \text{ is allocated at memory offset } \mu_j \}$$

A *Forensic Kernel Object Profile* (OP) is a set of elements that represent memory snapshots for memory allocated to the kernel object O_x through execution of a program represented by O_x. An element in the set is called *kernel object's memory snapshot* (α) at an executed instruction χ, and is defined as a set of 2-tuples (μ_ρ, v_ρ), where μ_ρ represents a memory offset for a kernel object property $\rho_x \in O$ and v_p represents a value assigned to property ρ_x as a result of an execution state χ, such that:

$$\chi_{cur}\,(\alpha) = \{(\mu_1 := v_1) \wedge\ (\mu_2 := v_2) \ldots \wedge \ldots (\mu_n := v_n)\}$$

Figure 2 presents an example of forensic kernel object profiling of the `EPROCESS` dynamic kernel object in Windows operating system kernel. The rounded boxes in Figure 2 show various *kernel object's memory snapshot* (α) at different executed instructions. For example, at the initialization state, the operating system initializes properties of a kernel object through assigning a process name, unique process id, initializing the process kernel information, determining control flags and assigning proper access security token at offsets, +0x154, +0x084, +0x000, +0x248 and +0x0c8, respectively [32]. Through program execution, properties of the kernel object are changed to allow the program to execute intended code, i.e. new security flags are assigned and existing control flags are updated, etc.
Thus, according to presented profiling method, different kernel object snapshots (α) for dynamic memory allocated to the object are acquired. For example, at instruction execution state χ_{cur+1}, characteristics of profiled kernel object are defined as:

$$\chi_{cur+1}\,(\alpha) = \{(0x000 := v_{cur+1}) \wedge (0x084 := v_{init}) \wedge .. (0x0c8 := v_{cur+1}) \ldots\}$$

and the Forensic Kernel Object Profile (OP), is defined as:

$$\text{OP}(\text{EPROCESS}_{\text{malware}}) = \{\chi_{init}(\alpha)\ ,\ \chi_{cur}(\alpha)\ , \chi_{cur+1}(\alpha), \ldots, \chi_{lst}(\alpha)\}$$

Fundamentally, profiled memory snapshots encode changes in the object properties at different execution states, and determine characteristics of profiled object at every

execution state. The observed changes in monitored object properties, results in a unique property updates pattern that allows for development of a malicious kernel object profile and assists in differentiating malicious kernel objects from benign kernel objects.

To accurately profile properties of a specific kernel object, some properties in kernel object definition may include host or user specific data, e.g. timestamp of the object creation or user directory of downloaded malicious programs, etc. Such information is specific to the analysis environment configuration and may contribute to inaccurate profiles. Thus, kernel object properties of interest – and that are considered in profiling process – are properties that affect program execution in the operating system kernel and, if tampered with, monitored program may produce unpredictable behavior [33, 38]. Thus, user or host specific information is defined as a set of properties ρ_{ex} and are excluded from profiling process. Hence, a final profile denoted as $\mathrm{OP}(\mathrm{Obj_x})$ is formalized as follow:

$$\mathrm{OP}(\mathrm{Obj_x}) = \bigcup \chi_m(\alpha) - \bigcap_{\rho_{ex} \in \rho} \rho_{ex}$$

This formula represents the process of profiling a memory snapshot of a kernel object of interest at different instruction execution states. To generalize developed profiles and exclude properties that may produce false negative results, user specific information such as, process id, user timestamps or an executable location, are eliminated from profiling process and kernel specific properties are only considered in the profiling procedure.

3.3 From Malicious Code Execution to Object Profiles

As previously illustrated, memory regions allocated to a malicious kernel objects are profiled to determine the object characteristics in different execution states. This section presents a formalization of code execution states that stimulate presented profiling process. A malicious executable *P* is modeled as a binary program that holds a set of assembly language instructions $I = \{I_1, I_2 \ldots . I_n\}$. The execution of the malicious program *P* possesses a sequential execution of instruction set *I* in *P* with an exception to instructions that change program control-flow, e.g. jump instructions. Consequently, malicious program execution can be presented as a *control flow graph* (CFG) [34, 39]. CFG (P) can be defined as 2-tuple (S, E), where $S \in \Sigma$ is an execution state presented as assembly instructions, and *E* is a set of edges $E \subseteq S \times S$, where *E* represents a transition corresponding to execution of a malicious instruction in memory.

The kernel object memory profiling procedure based on presented formalization is defined as 3-tuple, $P = [\![I, E, PR\langle I_g \rangle]\!]$, where:

- *I* is a set of execution states, each representing an instruction in determined malicious executable.
- *E* is a set of edges corresponds to transition to an instruction.
- $PR\langle I_g \rangle$ is the profiling procedure that acquire a snapshot of memory allocated to malicious kernel object at execution of an instruction I_g.

#	INS	Argument
1	push	dwDesiredAccess
2	call	ds:openMutex
3	cmp	[ebp+var_4] , eax
4	Jz	Short loc_30902E
5	push	offset Name
6	push	0
7	push	0
8	call	ds:createMutex
loc_30902E		
9	push	0
10	call	ds:exit
loc_30903A		
11	mov	esp , ebp
12	pop	esp
13	retn	

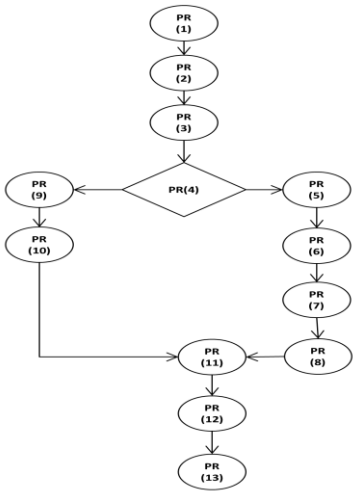

Figure 3: Executed Code CFG to Memory Profiles Snapshots CFG.

Figure 3 presents a practical example of modeling a malicious executable, and creating a profile of memory allocated to kernel objects in dynamic kernel memory. Code in Figure 3 is modeled as a CFG graph. Each instruction is modeled as an execution state, and execution of next instruction is represented as a transition in the CFG graph. Instructions similar to those presented in line 4 represents a branching transition of the CFG graph to instructions located at `Loc_30902E` in memory. Through execution of the malware CFG modeled graph, malicious kernel objects are profiled using presented method, and added to the malicious kernel object profiling space.

Intuitively, the object profile space can, as well, be modeled as CFG of profiles analogy to code CFG. Profile CFG illustrated in Figure 3 represents an `EPROCESS` object of a running malicious process in Windows operating system kernel. Consecutive memory snapshots are acquired for malicious object through malicious code execution in a managed environment. Acquired memory profiles are, then, used to determine malicious `EPROCESS` kernel objects in a digital forensic investigation of memory images that infected with profiled malware sample.

4 Implementation and Case Study

Implementing a prototype dynamic kernel object profiler for digital forensic investigation purpose requires dynamic access to memory regions allocated to malicious kernel objects and monitoring executed instructions by malicious code. That is, QEMU [35], an open source processor emulator was used to accomplish aforementioned requirements. QEMU was customized to allow instruction emulation to stimulate proposed kernel object profiling procedure. Note that, in this research,

Windows operating system kernel is approached for presented profiling process, specifically dynamic kernel objects that represent running process in Windows, such as, EPROCESS and its substructures: _KPROCESS and _KTHREAD. This is because EPROCESS kernel object is a common target for forensic investigators of malware and references different types of kernel objects that are essential to the investigation. For example, EPROCESS keeps track of memory allocated to a program through the Virtual Address Descriptor data structure, and files mapped in memory [36]. Determining memory regions allocated to a program in QEMU is accomplished through monitoring the value loaded into CR3 processer registers. This value represents the page-directory base register (PDBR) of physical memory address of current program's process loaded into QEMU processor [32]. Monitoring the aforementioned register enables determining the physical memory address of currently loaded EPROCESS into the emulation processer. Once memory region for an EPROCESS is determined, the memory region is mapped to the formal definition of EPROCESS as described in Windows operating system kernel specification to identify the offset of each property in monitored object and its value. Finally, the proposed profiling procedure snapshots identified memory offsets, as previously described, at invocation of emulated instructions in QEMU processor. Acquired malicious EPROCSS profiles are, then, used to automate identifying if a kernel objects is malicious or not in post-mortem analysis of a memory.

To automate the process of malicious EPROCESS extraction and identification, a plugin to Volatility Memory Forensic Framework [26] developed to automatically extract kernel objects and match extracted objects with developed malware profiles. If an extracted object matches a profile, memory regions allocated to the suspect program and referenced by the suspect EPROCESS are automatically extracted for further forensic analysis.

4.1 Zeus Toolkit Profiling Case Study

To evaluate the efficacy of the proposed method, a forensic profile for Zeus malware [37] was developed. Zeus is a toolkit that is commonly used to commit financial crimes on the Internet [37]. In the last few years, Zeus toolkit has become a dominant tool for cyber criminals since it allows to, easily, configure a malicious binaries to commit a variety of cybercrimes, such as stealing the users' Internet baking accounts and credit card information and leaking user-sensitive financial information to a black market.

To verify developed Zeus's profiles, four Windows 7 virtual machines infected with Zeus malware were deployed. Dynamic kernel memory of each infected VM acquired for analysis, and matched with developed Zeus profiles. Kernel objects in each forensic memory image have been processed using Volatility with developed extraction and identification plugin.

Zeus Variants	*Acquired memory Snapshots*	*# Benign Kernel Objects*	*False Detections*
ntos.exe	4511	64	-
oembios.exe	4009	52	-
Sdra64.exe	3794	52	-
PP08.exe	3401	43	-

Table 1: Results of Profiling the CFG Graphs Corresponds to Zeus's Executable

This allows automatic identification of malicious kernel objects related to Zeus, and also automatically extracted memory regions referenced by Zeus's `EPROCESS` kernel object.

Table 1 shows the results of Zeus's profiling process and characteristics of each acquired forensic memory image for investigation. As shown in Table 1, Zeus's Kernel Object Profile (OP) is consists of up-to 4500 object memory snapshots.
In essence, acquired memory snapshots of Zeus's kernel object correspond to emulated instructions of Zeus's executable and executed states in Zeus's modeled CFG graph, as previously described. In addition, each acquired memory image has up-to 60 `EPROCESS` kernel objects for commonly-used benign software e.g. Microsoft Internet Explorer, MS Media Player, and MS Office.
Matching extracted kernel objects with acquired profiles resulted in identification of Zeus's `EPROCESS` objects in all memory images without producing false positives with benign `EPROCESS` objects. Furthermore, to verify the preciseness of acquired profiles, Zeus's profiles have been used to investigate freely available [26] seven different Windows XP SP2 forensic memory images infected with different malware samples. Developed profiles, however, did not produce false results with other malicious kernel objects.

5 Discussion and Future Work

Although presented profiling method shows promising results in determining characteristics of malicious kernel objects and automating malicious kernel object identification in post-mortem memory analysis, some improvements are required.
The proposed method is considered a complementary approach for dynamic analysis techniques; thus, challenges to dynamic analysis approaches may, also, affect the proposed method. For example, to develop a complete object profile, all execution paths in malicious code's CFG graph have to be considered. Otherwise, if a malicious code has multiple execution paths, proposed method may results in incomplete profiles and may produce false results. Thus, the proposed method has to be assisted with proposed improvements to dynamic analysis approaches for digital forensic investigation.

Hence, our future work plan includes approaching proposed dynamic analysis improvements and implementing improved approaches in a forensic-specific malware investigation platform.

6 Conclusion

This research highlighted the limitations of employing dynamic malware analysis approaches in digital forensic investigations of malware, and proposed a set of improvements to presented limitations. Based on highlighted limitations, a method proposed to profile malicious kernel objects in dynamic kernel memory. Developed malware profiles allow investigators to automatically identify malicious kernel objects during post-mortem memory analysis of acquired dynamic kernel memory of the victim's computer. To allow an automated profiling of malicious kernel objects, a prototype malware sandbox solution developed and used to profile a malware family that is commonly used to commit finical crime on the Internet.

References

1. Yin, H., et al.: Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In: Proceedings of the 14th ACM conference on Computer and Communications Security, (2007).
2. Yin, H., Z. Liang, and D. Song.: HookFinder: Identifying and Understanding Malware Hooking Behaviors. In: Proceedings of Distributed System Security Symposium. (2008).
3. Kolbitsch, C., et al.: Effective and Efficient Malware Detection at the End Host. In: Proceedings of the 18th Conference on USENIX Security Symposium, (2009).
4. Vasudevan, A. and R. Yerraballi.: Cobra: Fine-grained Malware Analysis using Stealth Localized-Executions. In: Proceedings of IEEE Symposium on Security and Privacy, (2006).
5. Dinaburg, A., et al.: Ether: Malware Analysis Via Hardware Virtualization Extensions. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, (2008).
6. Lanzi, A., M. Sharif, and W. Lee.: K-Tracer: A System for Extracting Kernel Malware Behavior. In: Proceedings of The 16th Annual Network and Distributed System Security Symposium, (2009).
7. Bayer, U., et al.: Dynamic Analysis of Malicious Code. In: Journal in Computer Virology 2(1): p. 67-77, (2006).
8. Christodorescu, M. and S. Jha.: Static Analysis of Executables to Detect Malicious Patterns. In: Proceedings of the 12th USENIX Security Symposium, (2003).
9. Moser, A., C. Kruegel, and E. Kirda.: Limits of Static Analysis for Malware Detection. In: Proceedings of Computer Security Applications Conference, (2007).
10. Egele, M., et al.: A Survey on Automated Dynamic Malware Analysis Techniques and Tools. In: ACM Comput. Surv., 44(2): p. 1-42, (2012).
11. Farmer, D. and W. Venema.: Forensic Discovery. Addison-Wesley, (2005).
12. Nance, K., M. Bishop, and B. Hay.: Virtual Machine Introspection: Observation or Interference?. In: IEEE Security and Privacy, (2008).

13. Sharif, M., et al.: Impeding Malware Analysis Using Conditional Code Obfuscation. In: Proceedings of the Network and Distributed System Security Symposium, (2008).
14. You, I. and K. Yim.: Malware Obfuscation Techniques: A Brief Survey. In: Proceedings of the Int. Conf. on Broadband, Wireless Computing, (2010)
15. Balzarotti, D., et al.: Efficient Detection of Split Personalities in Malware. In: Symposium on Network and Distributed System Security (NDSS), (2010).
16. Moser, A., C. Kruegel, and E. Kirda.: Exploring Multiple Execution Paths for Malware Analysis. In: IEEE Symposium on Security and Privacy, (2007).
17. Shosha, F.A., J. James, and P. Gladyshev.: A Novel Methodology for Malware Intrusion Attack Path Reconstruction. In: International ICST Conference on Digital Forensics & Cyber Crime (ICDF2C), (2011).
18. Gladyshev, P. and A. Patel.: Finite State Machine Approach to Digital Event Reconstruction. In: Digital Investigation, (2004).
19. Forrest, S., S. Hofmeyr, and A. Somayaji.: The Evolution of System-Call Monitoring. In: Proceedings of the Annual Computer Security Applications Conference, (2008).
20. Mutz, D., et al.: Anomalous System Call Detection. In: ACM Trans. Information System Security, (2006).
21. Riley, R., X. Jiang, and D. Xu.: Multi-Aspect Profiling of Kernel Rootkit Behavior. In: Proceedings of the 4th ACM European Conference on Computer Systems, (2009).
22. Rhee, J., Z. Lin, and D. Xu.: Characterizing Kernel Malware Behavior With Kernel Data Access Patterns. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, (2011).
23. Malin, C., E. Casey, and J. Aquilina.: Malware Forensics: Investigating and Analyzing Malicious Code. Syngress, (2008).
24. Newsome, J. and D. Song.: Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In: Proceedings of Network and Distributed System Security Symposium (NDSS), (2005).
25. Schwartz, E., T. Avgerinos, and D. Brumley.: All You Ever Wanted to Know About Dynamic Taint Analysis and Forward Symbolic Execution. In: IEEE Symposium on Security and Privacy (Oakland'10), (2010).
26. Volatility.: An Advanced Memory Forensics Framework. Available from: https://www.volatilesystems.com/default/volatility, (2012).
27. Dolan-Gavitt, B.: The VAD Tree: A Process-Eye View of Physical Memory. In: Digital Investigation, (2007).
28. Schuster, A.: Searching for Processes and Threads in Microsoft Windows Memory Dumps. In: Proceedings of the 6th Annual Digital Forensic Research Workshop, (2006).
29. Marrington, A., et al.: A Model for Computer Profiling. In: The Third International Workshop on Digital Forensics, (2010).
30. Hoglund, G.: Rootkits: Subverting the Windows Kernel. Addison-Wesley, (2005).
31. Wang, Z., et al.: Countering Kernel Rootkits With Lightweight Hook Protection. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, (2009).
32. Russinovich, M.: Windows Internals. Microsoft Press, (2009).
33. Dolan-Gavitt, B., et al.: Robust Signatures for Kernel Data Structures. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, (2009).
34. Clarke, E., O. Grumberg, and D. Peled.: Model Checking. MIT Press, (2000).
35. Bellard, F.: QEMU, A Fast and Portable Dynamic Translator. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, (2005).

36. Van Baar, R.B., W. Alink, and A.R. Van Ballegooij.: Forensic Memory Analysis: Files Mapped in Memory. Digital Investigation, (2008).
37. Binsalleeh , H., et al.: On the Analysis of the Zeus Botnet Crimeware Toolkit. In: Proceedings of the Eighth Annual International Conference on Privacy Security and Trust, (2010).
38. Shosha, F.A., J. James, L. Chen-Ching and P. Gladyshev.: Evasion-Resistant Malware Signature Based on Profiling Kernel Data Structure Objects. In: Proceedings of the 7th Intl. Conference on Risks and Security of Internet Systems (CRiSIS), (2012).
39. Shosha, F.A., J. James, L. Chen-Ching and P. Gladyshev.: Towards Automated Forensic Event Reconstruction of Malicious Code. In: Proceedings of the 15th International Symposium on Research in Attacks, Intrusions and Defenses (RAID), (2012).