ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Kernel Level Malware Monitorer and Detector in Virtual Environment

[1]P. Ashok Kumar, [2]B.K. Prasath, [3]M. Nandhakumar and [4]A. Chitra

[1]Kingston engineering college, PG Student, Computer Science and Engineering Department, 632059. Vellore. Tamil Nadu, India.
[2]Kingston engineering college, PG Student, Computer Science and Engineering Department, 632059. Vellore. Tamil Nadu, India.
[3]Kingston engineering college, PG Student, Computer Science and Engineering Department, 632059. Vellore. Tamil Nadu, India.
[4]Kingston engineering college, Professor, Computer Science and Engineering Department, 632059. Vellore. Tamil Nadu, India.

ARTICLE INFO

Article history:
Received 12 March 2015
Accepted 28 April 2015
Available online 5 May 2015

Keywords:
OS kernel malware characterization, data-centric malware analysis, virtual machine monitor.

ABSTRACT

In the recent trends, Malware discovery and analysis approaches are focused in code-centric aspects of malicious programs. According to the current scenario, advanced tools are used in the methods of malware coding that includes reusing legitimate code or obfuscating malware code to circumvent the detection. Our projected approach dealt with the code-centric approaches by proposing a kernel malware characterization to detects, characterize and prevent the malware attacks supported the properties of knowledge objects manipulated throughout the attacks. This Approach postulates are a kernel object mapping technique in runtime that reads the kernel objects to identify the malware acquired based on the signature and patterns of the malware. The identified malware are prevented by a monitoring application that utilizes a byte based scanner. This approach has associate extended coverage that detects and prevents not only the malware with the signatures but also the malware attack patterns by modeling the low level knowledge access behaviors as signatures. Our experiments against a range of real-world kernel root kits demonstrate the effectiveness of malware signatures. Hybrid Malware detect memory Mapped provides an optimized solution to analyze windows kernel-level code and extract malicious behaviors from root kits, including sensitive data access, modification and triggers. A new technique provides a combination of patch making and memory mapping in kernel level. It will identify the malware influenced sensitive data and possible solution for this problem.

To Cite This Article: P. Ashok Kumar, B.K. Prasath, M. Nandhakumar and A. Chitra., Kernel Level Malware Monitorer and Detector in Virtual Environment. ***Aust. J. Basic & Appl. Sci.,*** *9(20): 432-436, 2015*

INTRODUCTION

Malware use a range of techniques to cause divergence within the attacked program's behavior and achieve the attacker's goal. In older days, malicious programs such as viruses, worms, and exploits are using code injection attacks that inject malicious code into a program to perform a wicked perform. Intrusion detection approaches are also considered to be of malware injections (Cowan, C., 1998; Seshadri, A., 2007). Alternate attack vectors were devised to avoid violation of code integrity and so elude such detection approaches. for example, return-to-libc attacks return-oriented programming and jump-oriented programming (Chen, P., 2009; Davi, L., 2010) employ existing code to make malicious logic. to boot, kernel malware will be launched via vulnerable code in program bugs third-party kernel drivers, and memory interface which may enable manipulation of kernel code and information victimization legitimate code (i.e., kernel or driver code).

This arms-race between malware and malware detectors centers on properties of malicious code based injection/integrity of code or the causative sequences of malicious code patterns. While the bulk of existing work focuses on the code malware executes, comparatively very little work has been done that focuses on the information it modifies. Data-centric approaches need neither the detection of code injection nor malicious code patterns. They are not directly submergible victimization code apply or obfuscation techniques. However, sleuthing malware supported knowledge modifications has a distinctive challenge that produces it distinct from code based approaches. Correspondingly, conventional integrity checking can't be applied to knowledge properties. additionally, observance knowledge objects of AN operational system (OS) kernel has extra challenges as a result of An OS may be the bottom software package layer in typical computing environments, which means that there's no observance layer below it (Sharif, M., 2009).

Corresponding Author: P. Ashok Kumar, Kingston engineering college, CSE Department, 632059.Vellore, TamilNadu, India.

Australian Journal of Basic and Applied Sciences, 9(20) June 2015, Pages: 432-436

Malware observe memory Mapped provides an optimized answer to investigate windows kernel-level code and extract malicious behaviors from root kits, as well as sensitive knowledge access, modification and triggers. A new technique that provides a mix of backward slicing choice to check the mapped memory by slicing. It'll establish the malware influenced sensitive knowledge and potential answer for this drawback. In this paper, the malware in the virtual machine is being detected and also tends to be monitored with the help of malware detector.The Monitoring application execution involves Memory Management Leaks, Memory Performance Checks, Unmanaged Code execution, Listing down the malware and fixing it by implementing over some testing analysis like Malware bytes Anti-Malware (MBAM) scanner. Dynamic detection of malware activity in virtual environment detects the vulnerable activity in kernel aided with proof carrying out over the injected malware code and memory leakage mechanism. Malware detector and malware monitoring will eradicate and prevent the vulnerable malwares in the virtual machine by providing anti-malware protection.

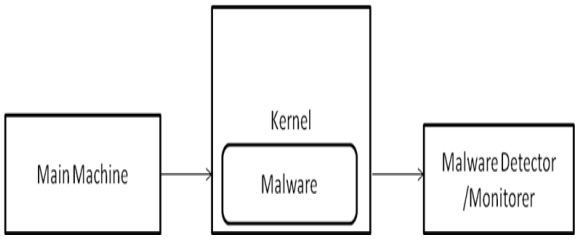

Fig. 1: Kernel Malware Monitorer.

Problem Description:

In our planned approach, the malware within the virtual machine is being detected and additionally tends to be monitored with the assistance of malware detector in its sequence of scenarios such as Demo Malware creation Process carried out by Main machine app caller that invokes the Monitored. The ministered manages the memory by checking the memory leaks and validates the invalid file properties to apply patching concept to overcome the vulnerabilities acquired by the malware.

The main components considered in the monitorer/Detector zone of kernel are

- ***Memory Mapping / Leaks:***

Memory mapping / leaks approach recognizes data objects based on memory allocation with inter-memory pointers. A firmware passes the instruction to the OS kernel about mapping the data that already constitute of necessary information about the memory. Memory leak is considered to be the unaccessed memory stored in the data object.

- ***Memory Access / Irregular Memory wastage:***

Kernel consider the fall in unavailable physical address range and kernel code with the initialized data structures as reserved According to the memory request such as static or dynamic the frames are allocated by the kernel as the kernel doesn't trust as the user mode as save one. A memory gets allocated in situations such as 1) new process is created 2) A running process decides to load an entirely different program (using exec). In this case, the process ID remains unchanged, but the memory regions used before loading the program are released and a new set of memory regions is assigned to the process 3) A running process may perform a "memory mapping" on a file 4) A process may keep adding data on its User Mode stack until all addresses in the memory region that map the stack have been used. In this case, the kernel may decide to expand the size of that memory region 5)A process may create an IPC-shared memory region to share data with other cooperating processes. In this case, the kernel assigns a new memory region to the process to implement this construct 6) A process may expand its dynamic area (the heap) through a function such as malloc. As a result, the kernel may decide to expand the size of the memory region assigned to the heap [ref: Wiki Direct].The Accessibility and unallocated pipelined frames are identified.

- ***Byte Code Instruction Checks / Improper properties of exe:***

Byte code, also called p-code (portable code), is a form of instruction set for well-organized execution by a software interpreter. The monitorer checks the interpretation and the mal functioning properties of executable files of any application on the OS.

The fig2 depicts the overall structural design of the projected approach. A Virtual Machine and Main machine acts in the design in order to differentiate the malware detection clearly. An application request from the Virtual machine to the kernel Mode Service is raised initially. The Kernel constitutes of Malware and infected host executable. The service provision from the kernel mode to the requestor such as user or VM will be monitored by Malware Detector/Monitorer.

Fig 3 depicts the functional Design of the kernel Moniterer, The characterization of malware are

represented in this functional design. There are three zones divided in the functional architecture such as Virtual Machine zone, Main machine Zone and the monitorer zone. The Virtual Machine undergoes generic problems such as File handling issue and File size Variation issue. The File handling issue is that an application crossely gets opened in the form of another application, Say for example; A PDF file opens as VLC Media Player. The File size Variation issue is an abrupt change in file size due to the malfunctioning of the malware. The monitorer application checks the Memory Performance with respect to kernel and application level, Memory management such as memory allocation and deallocation for writes and reads in the kernel and application level are checked, Interoperating unmanaged code evaluation that acts as communication chamber between the managed and unmanaged code utilized in the kernel and application level. The arise of issue in these memory and managed code concepts are sorted out to apply in the Malware bytes anti-malware scanner. Malware bytes anti-malware scanner is the proposed algorithm, it utilizes the behavior of the malware and identifies the signature of the malware with a set of kernel versus main machine association rules. The Main machine Zone posses the Virtual Machine zone where the memory of the main and the virtual machine are shared according to the usage. The Memory compares check and application performance check of the monitorer zone evaluates the change in the memory after the revamping process done in the file in kernel level.

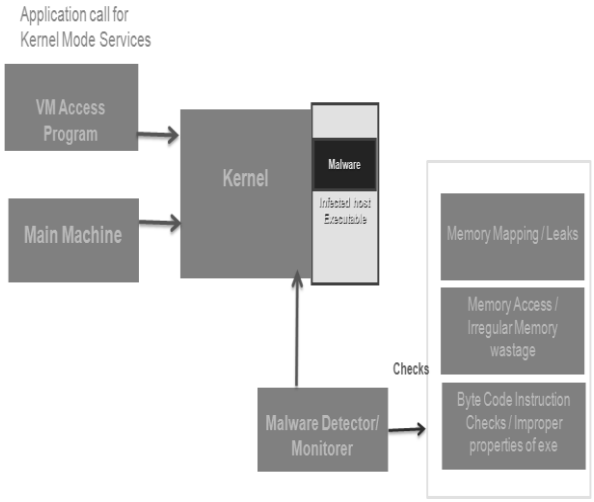

Fig. 2: Kernel Monitorer Architecture.

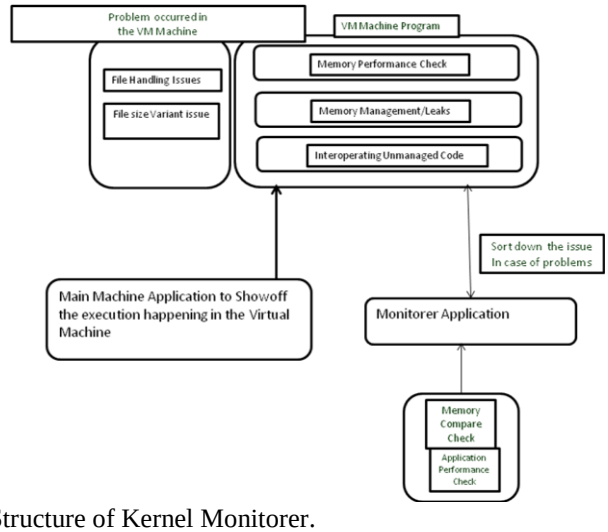

Fig. 3: Functional Design Structure of Kernel Monitorer.

In the projected scenarios, initially the concept of Malware Creation is done; it will be taken place in the virtual machine where the process of sophistication malware will affect the virtual environment. A script is written that creates improper properties of exe files, memory leakages, and other similar issues. A system call is a mechanism that is used by the application program to request a service from the operating system. The malware program will enable the operating system to interact with a hardware device. The kernel takes responsibility for deciding many running programs

In the main machine app calling module. User program running under guest OS will create issues in kernel call instruction. When guest OS(virtual os) returns from system call, the monitoring mechanism will be done by the virtual memory management (VMM) to invoke the malware. The subset facilities of the underlying machine will invoke and monitor the memory management with extra mechanisms implemented by the operating system. The memory leaks in virtual machine by the malware will be identified in order to analyze the impact in the main machine. The memory leak check module will also identify the other malfunctionalities (improper file handling) that occur in the virtual environment. The control of kernel operations will be mapping up with the overall machine memory in order to optimize the use of RAM, where there is a Memory Management issue. The Memory Manager includes memory-mapped files. Memory-mapping can speed-up sequential file processing due to the fact the data is not sought randomly, and it provides a mechanism for memory-sharing between processes. Scheduling of computing time and memory management is also part of the virtual machine monitors responsibilities. The invalid file properties of the malware will be monitored and identified to rectify the issues in the main machine. Validating the invalid files will be done with the help of the malware detector monitoring mechanism to avoid the misleading of file activities. A patch is a small text document containing a delta of changes between two different versions of a source. Patches are created with the `diff‟program in the kernel (Seshadri, A., 2007). The Patches for the kernel are generated relative to the parent directory holding the kernel source dir. The Monitorer uses the patch file to revamp the issues driven in the virtual and main machine, it analyses the changes made in the file in order to refix the affected file as it was before.

DKOM –Direct Kernel Object Malware Algorithm:

This technique was presented for the malware manipulation process in the existing papers. The object malware algorithm detects the system and enables out-of-the box, tamper-resistant malware detection without losing the semantic view (Sharif, M., 2009). In general this algorithm prevents the system comprises at least one guest operating system and at least one virtual machine, where the guest operating system runs on the virtual machine. Having virtual resources, the virtual machine resides on a host operating system. The virtual resources include virtual memory and at least one virtual disk where it acts to prevent the malware .DKOM along with a virtual machine inspector, a guest function extrapolator, and a transparent presenter, the virtual machine examiner resides outside the virtual machine. The demerits of this algorithm are that the identification factors done with the objects cannot be utilized in the patch applying methodology that fixes the malware issue (Riley, R., 2009).

Malware bytes anti-malware scanner algorithm:

The Malware anti-malware scanner algorithm is configured to use the interpreted virtual memory states and the interpreted virtual disk states to detect system's malware and the affected files. The instructions executed from outside of the virtual machine, comprising files to retrieve improper exe in the virtual machine's internal. Based on non-intrusive virtual machine introspection without perturbing their execution, the virtual resources extrapolating guest functions by interpreting the virtual memory states and the virtual disk states. This algorithm gets the malware behavior with association functions in dynamic execution, It Utilizes a multiple kernel runs in the signature generation stage.

Let us consider for a malicious kernel run Km for the data TD with malware M is $D_{M,Km}$ and $D_{M,Vm}$ represents a data behavior profile for a VM kernel execution. We apply set operations on n malicious kernel runs and m m kernel runs as follows. The generated signature for the behavior of data is

$$S = \bigcap_{Km\in[1,n} TD_{M,Km} - \bigcup_{Vm\in[1,m} TD_{M,Vm}$$

This formula represents that SM is the set of data behavior that consistently appears in n malware runs, but never appears in m kernel runs. The underlying observation from this formula is that kernel malware will consistently perform malicious operations during attacks. This means, we can estimate malware behavior by taking the intersection of malicious runs. In general, Data characterizes the malware behavior by using dynamic kernel execution. The malware behavior is The likelihood (LM) that a malware program in a tested run (Tr) is defined by deriving a set of data behavior elements E which belong to the data behavior profile (P) i.e., E∈ P. This set I corresponds to the intersection of E and P (i.e., $I = \{i|i \in E \wedge i \in P\}$).

Conclusion:

As a perceptive comprehensive algorithm Malware bytes anti-malware scanner algorithm for Detecting the malware occurred abruptly in the system or in the virtual machine of the system. Monitorer can obtain a high-quality resultant in the area of malware detection and fixation. The problem in existing system for providing solution to the detected malware is fixed in this paper. The generated monitorer algorithms in the postulates have the analysis in the form behavior and signature of the malware under study. Our Study and postulates with improved set of formulae found to be optimal option than the existing scenarios. This is experimentally proved for the effectiveness in the

Kernel Level Malware monitorer and Detector in Virtual Environment.

REFERENCES

Cowan, C., C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, *et al*., 1998. "StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks," in Proc. 7th USENIX Sec. Conf., pp: 63-78.

Seshadri, A., M. Luk, N. Qu and A. Perrig, 2007. "SecVisor: A tiny hypervisorto provide lifetime kernel code integrity for commodity OSes," in Proc.21st SOSP, pp: 1-17.

Chen, P., H. Xiao, X. Shen, X. Yin, B. Mao and L. Xie, 2009. "DROP:Detecting return-oriented programming malicious code," in Proc. 5th ICISS, pp: 163-177.

Davi, L., A.R. Sadeghi and M. Winandy, 2010. "ROPdefender: A detectiontool to defend against return-oriented programming attacks," Syst. Sec. Lab., Tech. Univ. Darmstadt, Darmstadt, Germany, Tech. Rep. HGI-TR-2010-001.

Sharif, M., A. Lanzi, J. Giffin and W. Lee, 2009. "Automatic reverse engineering,of malware emulators," in Proc. 30th IEEE Symp. Sec. Privacy, pp: 1-16.

2001,. Linux on-the-Fly Kernel Patching Without LKM [Online]. Available: http://www.phrack.com/issues.html?issue=58&id=7

Sharif, M., A. Lanzi, J. Giffin and W. Lee, 2009. "Automatic reverse engi- neering of malware emulators," in Proc. 30th IEEE Symp. Sec. Privacy, pp: 1-16.

Riley, R., X. Jiang and D. Xu, 2009. "An architectural approach to preventing code injection attacks," IEEE Trans. Dependable Secure Comput., 7(4): 351-365.

Etoh, H., 2011. GCC Extension for Protecting Applica- tions From Stack-Smashing Attacks [Online]. Available: http://www.trl. ibm.com/projects/security/ssp/.

Seshadri, A., M. Luk, N. Qu and A. Perrig, 2007. "SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes," in Proc. 21st SOSP, pp: 1-17.