International Journal of Advanced Research in

Computer Science Engineering and Information Technology

Volume: 4 Issue: 3 18-Apr-2015,ISSN_NO: 2321-3337

FUSION MALWARE PERCEIVE REMINISCENCE MAPPER IN CORE CENTRIC VM CENTRED GREEN COMPUTING DOMAIN

G.Suganya B.Tech., S.Sharmila M.Tech.,

Student, Department of Information Technology, Vel Tech Multi Tech
Dr. Rangarajan Dr.Sakunthala Engineering College, India1.
Assistant Professor, Department of Information Technology, Vel Tech Multi Tech
Dr. Rangarajan Dr.Sakunthala Engineering College, India2.

.

ABSTRACT- *Kernel malwares can provide user level-malware characteristics with additional probabilities of hiding their malicious activities by altering the legitimate kernel behaviour of an operating system. Many research proceeds on malware hooking behaviour and defence and preventive actions for the same. Still an automated analysis of the actual malicious goals and patching the behaviour is not investigated properly. Fusion malware perceive reminiscence mapper provides an optimized solution to analyse windows kernel-level code and extract malicious behaviours from root kits, including sensitive data access, modification and triggers. A new technique which provides a combination of backward slicing option to check the mapped memory by slicing step by step in the kernel level. It will identify the malware influenced sensitive data and possible solution for this problem.*

Keywords— Kernel, malicious,triggers.

1 INTRODUCTION

Network Security is the process of taking physical and software preventative measures to protect the underlying networking infrastructure from unauthorized access, misuse, malfunction, modification, destruction, or improper disclosure, thereby creating a secure platform for computers, users and programs to perform their permitted critical functions within a secure environment. A new technique that provides a mix of backward slicing choice to check the mapped memory by slicing step by step within the kernel level. It will establish the malware influenced sensitive knowledge and doable resolution for this drawback.

Malware observe memory mapper 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. 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.

2 SYSTEM ANALYSIS

2.1 Existing System

The existing system goes along with the malware identification pattern using the obtainable hardware Services (kernel) that identifies by detecting the Malware in order to clean up the worms and viruses by providing temporary protection to system. Existing approach projects towards Memory Performance Check, Memory management Leaks and Interoperability between managed (New version of Microsoft Language like c#) and unmanaged code(Older version of Microsoft Language like VC++). System call through Windows level coding invokes some of the malicious malware specification matching up with the suspicious system calls arises with existing malicious activity in the virtual Operating System . Memory mapping/leaks leads to memory leakage in the virtual machine which leads to handle the files with the improper usage of Application call for Kernel mode Services. Irregular memory wastage and improper properties of exe files while accessing the VM access program are considered to be some of the drawbacks of the existing approach.

2.2 Proposed System

In our proposed approach, the malware in the virtual machine is being detected and also tends to be monitored with the help of malware detector. Monitoring application execution involves Memory Management Leaks, Memory Performance Checks, Unmanaged Code execution and Listing down the malware and fixing it by implementing over some testing analysis like ***Malware Bytes Anti-Malware*** (MBAM) scanner which was considered to be projected in our proposed analysis. 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.

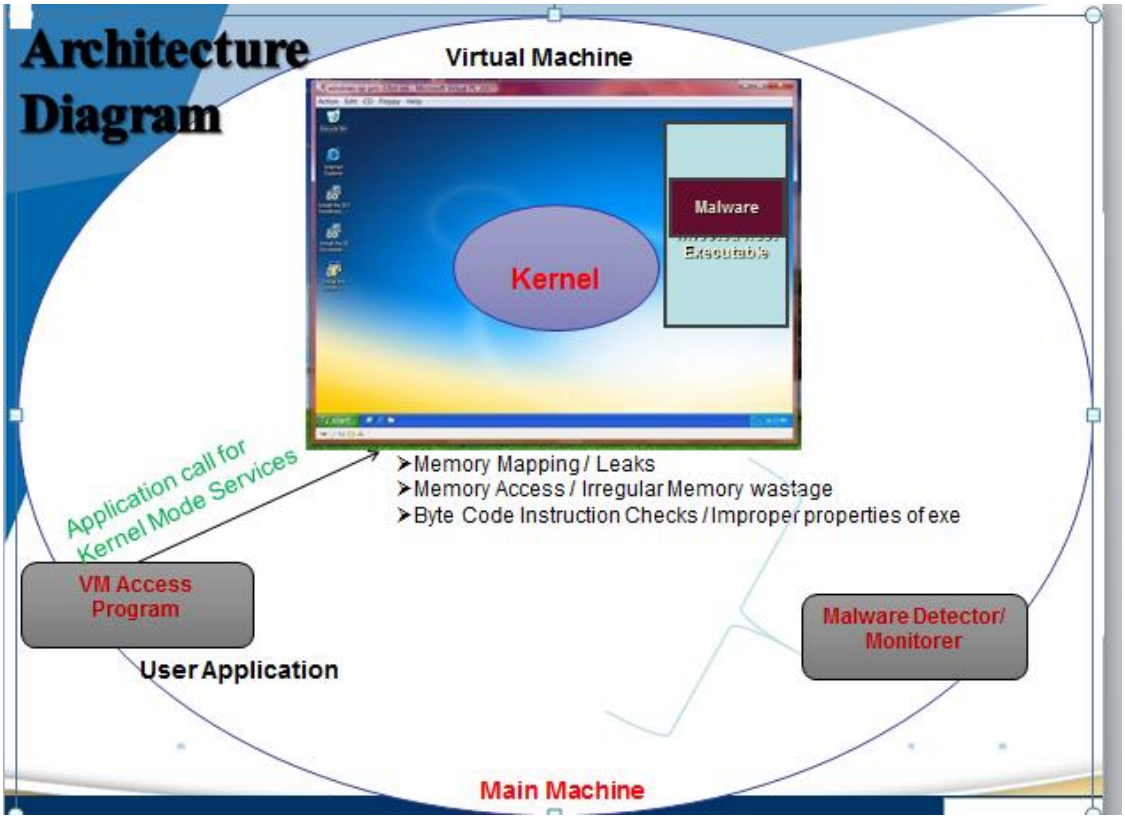
3 ALGORITHM

3.1 DKOM (Direct Kernel Object Manipulation):

This technique was presented for the malware manipulation process .The object malware algorithm detects the system and enables out-of-the box, tamper-resistant malware detection without losing the semantic view. 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 .

3.2 Malware Bytes Anti-Malware Scanner

The Malware Bytes 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.



4 IMPLEMENTATION

4.1 Malware creation module:

In this module, the malware creation will be taken place in the virtual machine where the process of sophistication malware will affect the virtual environment. The malware created in the virtual machine will create improper properties of exe files, memory leakages and other issues .In order to check these malware issues, at first no analysis of the core file will be done.

4.2 Main machine app calling module:

A system call is a mechanism that is used by the application program to request a service from the operating system. In this module 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.

4.3 Monitor invoke 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. In this module the subset facilities of the underlying machine will invoke and monitor the memory management with extra mechanisms implemented by the operating system.

4.4 Memory leak check module:

In this module 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 malfunctions (improper file handling) that occur in the virtual environment.

4.5 Memory management mapping module:

In this module 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 monitor's responsibilities.

4.6 Invalid file properties validation module:

In this module 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.

4.7 Patch applying module:

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. In this patch applying module, the patches for the kernel are generated relative to the parent directory holding the kernel source dir.

4.8 Performance evaluation module:

In this module we will evaluate the performance of the malware monitoring system. The performance evaluation module will eradicate the overall performance of the virtual machine and the main machine after the malware were fixed and rectified.

5 CONCULSION AND FUTUREWORK

The system works by building a live kernel object map which can reliably detect data hiding rootkit attacks due to its unhampered view of kernel objects. We propose a replacement malware signature approach victimization consistent patterns specific to malware attacks. Malware detector and malware observation can eradicate and stop the vulnerable malwares within the virtual machine by providing anti-malware protection. The map is then used in combination with a monitoring agent to track memory access patterns on kernel data objects. Based on these access patterns, we propose a new malware signature approach using consistent patterns specific to malware attacks. We demonstrate this scheme is not only effective at detecting previously evaluated rootkits but also their variants .we envisions cell-phones equipped with sophisticated sensors that stream data to a centralized service for processing. Traditional operating systems have a fundamental mismatch with such networked stream-oriented applications. A traditional OS has a process-centric view. The job of a traditional OS is to run user tasks, not store and serve data.

We propose a different approach in which the data-centric semantics of an application are exposed to the operating system. Our system, called Data-centric OS, enables a model where the application can simply declare its monitoring semantics by instantiating a Query File. Once an application has created a Query file, the application is no longer in the critical path of filtering, processing or storing a network stream. Instead, the Query File System plays an active role in efficiently processing and storing a network stream according to application semantics and the exposing application-level semantics to the OS, helps to filter unimportant data early and up calls to user-space are made only for semantically meaningful data. Application-specific processing paths can be set up to efficiently process incoming network packets. Also, OS resources may be allocated and adapted with varying data rates according to application semantics and different operations on the same data can be integrated.

REFERENCES

[1] M.Abadi, M.Budiu, Ú.Erlingsson and J.Ligatti, “Control-flow integrity:
Principles, implementations and applications,” in *Proc. 12th ACM Conf. CCS*, Nov. 2005, pp. 1–4.

[2] A.Baliga, V.Ganapathy and L.Iftode, “Automatic inference and enforcement of kernel data structure invariants,” in *Proc. 24th ACSAC*, Dec. 2008, pp. 77–86.

[3] D.Balzarotti, M.Cova, C.Karlberger, C.Kruegel, E.Kirda. and G.Vigna, "Efficient detection of split personalities in malware," in *Proc. 17th Annu. NDSS*, Feb. 2010, pp. 1–17.

[4] U.Bayer, P.Milani Comparetti, C.Hlauscheck, C.Kruegel, and E.Kirda, "Scalable, behavior-based malware clustering," in *Proc. 16th Symp. NDSS*, Feb. 2009, pp. 1–26.

[5] F.Bellard, "QEMU: A fast and portable dynamic translator," in *Proc. USENIX Annu. Tech. Conf.*, Mar. 2005, pp. 41–46.

[6] E.Buchanan, R.Roemer, H.Shacham, and S.Savage, "When good instructions go bad: Generalizing return-oriented programming to RISC," in *Proc. 15th ACM Conf. CCS*, Oct. 2008, pp. 27–38.

[7] J.Butler.(2012, Dec. 12). *DKOM (Direct Kernel Object Manipulation)* [Online]. Available: http://www.blackhat.com/presentations/winusa- 04/bh-win-04-butler.pdf

[8] (2010). *Bypassing Non-Executable-Stack During Exploitation Using Return-to-Libc* [Online]. Available: http://www. citeulike.org/user/rvermeulen/author/Contex

[9] M.Carbone, W.Cui, L.Lu, W.Lee, M.Peinado, and X.Jiang, "Mapping kernel objects to enable systematic integrity checking," in *Proc. 16th ACM Conf. CCS*, Nov. 2009, pp. 555–565.

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