Advances in Science, Technology and Engineering Systems Journal Vol. 3, No. 1, 25-29 (2018)

www.astesj.com

Special issue on Advancement in Engineering Technology

ASTESJ

ISSN: 2415-6698

Virtual Memory Introspection Framework for Cyber Threat Detection in Virtual Environment

Himanshu Upadhyay*, Hardik Gohel, Alexander Pons, Leo Lagos

Applied Research Center, Florida International University, Miami, 33172, United States

A R T I C L E I N F O

Article history:
Received: 01 November, 2017
Accepted: 13 December, 2017
Online: 18 January, 2018

Keywords :
Cybersecurity,
Virtualization,
Linux threat detection,
Hypervisor,
Feature selection technique,
Memory forensic analysis,
Virtual machine introspection

A B S T R A C T

In today's information based world, it is increasingly important to safeguard the data owned by any organization, be it intellectual property or personal information. With ever increasing sophistication of malware, it is imperative to come up with an automated and advanced methods of attack vector recognition and isolation. Existing methods are not dynamic enough to adapt to the behavioral complexity of new malware. Widely used operating systems, especially Linux, have a popular perception of being more secure than other operating systems (e.g. Windows), but this is not necessarily true. The open source nature of the Linux operating system is a double edge sword; malicious actors having full access to the kernel code does not reassure the IT world of Linux's vulnerabilities. Recent widely reported hacking attacks on reputable organizations have mostly been on Linux servers. Most new malwares are able to neutralize existing defenses on the Linux operating system. A radical solution for malware detection is needed – one which cannot be detected and damaged by malicious code. In this paper, we propose a novel framework design that uses virtualization to isolate and monitor Linux environments. The framework uses the well-known Xen hypervisor to host server environments and uses a Virtual Memory Introspection framework to capture process behavior. The behavioral data is analyzed using sophisticated machine learning algorithms to flag potential cyber threats. The framework can be enhanced to have self-healing properties: any compromised hosts are immediately replaced by their uncompromised versions, limiting the exposure to the wider enterprise network.

1. Introduction

Dependency on computer systems has been growing exponentially in recent years. Government offices and business organizations are major targets of malicious actors for stealing highly valuable data. These entities are major Linux adopters nowadays because of its reputed safety and security. They are attempting to protect themselves from cyberattacks with digital defense techniques like encryption, firewalls and heuristic or signature scanning packages. Meanwhile, the number of attacks that involve infiltrating military data centers, targeting power grids, and stealing trade secrets from both private and public organizations continues to increase. The detection, response and reporting of these kinds of intrusions as well as other incidents involving computer systems, are crucial for cybersecurity professionals. This paper proposes a Virtual Memory Introspection based framework for detection, analysis and monitoring of malware behavior using memory forensics, during cyberspace attacks in a virtualized environment. The framework provides advanced instrumentation tools for control and monitoring of malware, fine-grained introspection of operating system Kernel and user process behavior using the well-known LibVMI technology, and analysis of behavioral data using state-of-art machine learning techniques. In study of research literature, one of the major methods of malware detection that has emerged over the years is Linux memory forensics [1] [2]. A number of authors have described novel detection systems using Memory Forensics, or using kernel data structure invariants as a reference frame to identify rootkit intrusions [3] [4]. The goal of this test technology is to facilitate threat assessment of malware, to understand its goals, and degrade impact on the compromised systems [5].

Further, as cyber-attacks continues to expand and the sophistication of the adversaries grows, defenders must adapt quickly in order to survive. There are numerous kinds of malware prevalent today, including Backdoors, Bots, Downloaders and Droppers, Ransomware, Rootkits, Scareware, Spyware, Exploits, Logic Bombs, Trojans, Viruses, Worms, etc. The detection,

[*] Dr. Himanshu Upadhyay, Email: upadhyay@fiu.edu

response and reporting of these kinds of intrusions as well as other incidents involving computer systems, are crucial for cybersecurity professionals. Present static methods used by anti-viruses are insufficient to stop advanced malware threats, which are capable of disabling the anti-viral software via root-kit mechanisms.

2. Virtualization & Data Analytics Framework

The proposed framework has the following major components as shown in Figure1:

2.1. Virtualization

The virtual machines, which are monitored for malware attacks, are collectively called the system under test (SUT). This system is hosted on a Xen hypervisor based virtualization platform. The behavioral data on the guest VMs is captured through Virtual Memory Introspection (VMI) using the LibVMI framework

2.2. Data Analytics

This platform consists of various traditional machine learning algorithms used to train the models and perform prediction using various test vectors consisting of malware and rootkits on the virtual machines (SUT). In this proposed framework, the authors are building an advanced data analytics platform on the database server, R runtime with in-memory analytics providing the scale and performance. Various traditional machines learning algorithms like Random Forest, Support Vector Machines, Logistics regression etc. will be utilized to perform Data Analytics. This platform analyzes the memory data structure captured on the VMI framework using cutting-edge machine learning algorithms. These algorithms are used to build the model to predict the malware behavior and display the results.

2.3. Test Control Center

The Test Control Center is an application that provides control and administration of the entire framework with an intuitive web based interface. The operator can create virtual machines (SUT), install benign applications and malware, capture behavioral information and use the Data Analytics platform to build the model and test virtual machines for malware using various traditional machine learning algorithms. Test Control Center is a centralized application to manage the test bed and execute various test cases. In the proposed framework, the authors are developing various modules like virtual machine management, network map, test case management, model management, configuration, testbed administration and help.

The three platforms working together will enable an organization to detect malware almost in real time and monitor its behavior in a virtual environment. The user provides a set of well-defined inputs to the virtualization platform based on predetermined test cases, and captures the kernel data structures information and transfers it to the Data Analytics platform. The Data Analytics platform uses traditional machine learning algorithms to build the model and predicts results based on the information extracted from guest virtual machine memory and displays the results on the Test Control Center. The following diagram in Figure 1 provides the overall system design of the framework:

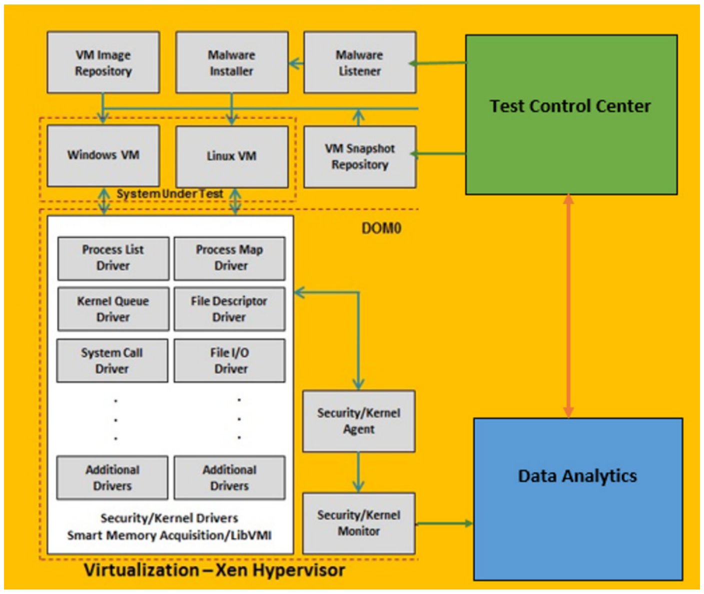

Figure 1: Virtualization & Data Analytics Framework System Diagram

The various components of the virtualization platform are discussed below.

I. VM Image Repository

Images of the different virtual machines will be stored in this location. It provides the user the ability to select the virtual machine images that will create the virtual machines for SUT with different configurations.

II. Malware Listener and Installer

This module listens to the command from the malware management module and communicates with the malware installer on the virtual machine to install the selected malware.

III. VM Snapshot Repository

This module will store the snapshots of normal operation which will later be used to roll back to the clean state after the test execution.

IV. Multiple Linux VMs

These are the guest virtual machines created by the operator based on predefined configurations.

V. Kernel Agent

Security/kernel agent manages all the drivers to extract the kernel data structures from the SUT. They are deployed on DOM 0 which is the secured location on the hypervisor with required privileges.

VI. Kernel Drivers

These the individual drivers which are used for smart memory acquisition of the kernel data structures from the SUT. These drivers use the LibVMI libraries for virtual memory introspection from the Dom0 level to extract kernel data structures. They are managed by Security/Kernel Agent.

VII. Kernel Monitor

This module communicates with data analytics platform and sends the data extracted by the Security/kernel agent to the listener on the data analytics platform to store the data in to the database.

VIII. Kernel Agent Listener

The Agent Listener plays a crucial role in the virtualization framework. This module communicates with the security/kernel monitor in the virtualization framework, and receives the extracted data from the security/kernel agent and stores it into the central database.

3. Virtual Memory Introspection (VMI) Framework

The VMI platform of this framework is built on the "Xen Hypervisor" which enables the operator to create multiple virtual machines with different configuration and perform system testing with the environment. The Linux machines are used as SUT, to be monitored for malware intrusion are hosted on this hypervisor. Xen hypervisor [6] provides two security domains for the hosted virtual machines. The virtual machine running under Dom0 (Domain 0) on the Hypervisor controls the resource allocation for the virtual machines that runs the Linux Guest operating system [7].

The major key objectives of virtualization framework are listed below:

I. Develop innovative ways to detect and monitor malware in a virtualized testbed with smart memory acquisition.

II. Deployment of virtualized environments along with advanced tools developed through this research for control and monitoring of the cyberspace test environment.

III. Cyber-attack emulation through infection and propagation of simulated endpoints.

IV. Virtual machine introspection, data collection, and monitoring of various aspects of the infrastructure through a centralized system.

V. Display results on the Test Control Center to monitor the impact of malware on SUT.

The SUT can be Linux or Windows guest virtual machines. Using the Test Control Center, the operator is able to create guest VMs on the Xen hypervisor from a virtual machine repository, introduce pre-determined malware into them, and capture kernel data structure information and stores them into the central database. We have developed an application called "Kernel Agent" that runs on the Dom0 virtual machine, and uses kernel drivers to perform smart memory acquisition on the guest operating system. This kernel information, which includes process details, memory data structures, and file system information, is written to a central database for data analytics.

Xen hypervisor is used to host, configure, and control the guest Linux virtual machine that will be tested for malware. The introspection is carried out through the "LibVMI" framework in conjunction with Google's "Rekall" profiles. The Rekall profile is a JSON file that contains the address mappings of all Linux Kernel data structures. By integrating LibVMI with Google Rekall, the process of extracting Kernel configuration parameters and data structures at run-time can be automated. This facilitates the smart memory acquisition of process behavioral data from the kernel. We are using Google's "Go" programming language for writing the introspector application, and to push the acquired data to the central database. The Go programming language can integrate with suitable C/C++ libraries, and significantly reduces the time needed for development and testing new build of each module. It provides better control over distribution and parallelization mechanisms of the Security Agent. We are using "Libvirt" library for management of the guest virtual machines through a custom built Introspector. Introspection requests are serviced by the LibVMI library, while the Libvirt library is used to facilitate the creation, starting, stopping, pausing and resuming of the guest Linux virtual machines [8].

The Introspector is comprised of several core subsystems that are necessary for lifecycle management and introspection of virtual machines on a host. The Introspector also maintains two socket connections that listen to requests for introspection or virtual machine administration coming from the Test Control Center.

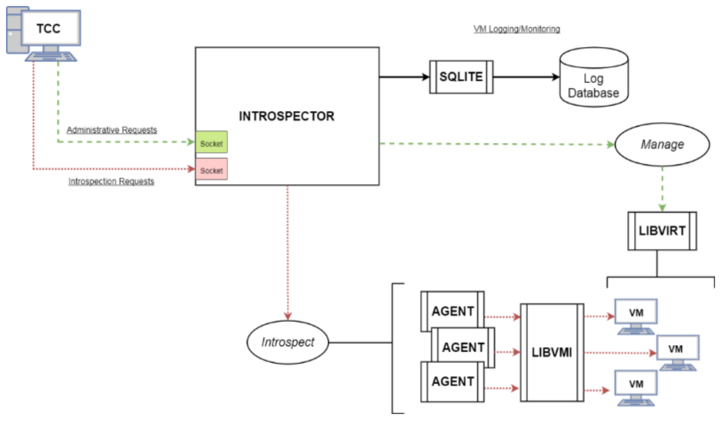

Figure 2: Proposed Virtual Machine Introspection Framework

Introspector: startup

Introspector (introspector.go) builds a Settings object (settings.go) by reading the values in the introspector.conf settings file. The configuration file is used to determine which IP/Port combination to use for handling virtual machine administration requests and introspection requests. The Introspector will also start the StateManager that manages a SQLite database containing the current state of the virtual machines in the system.

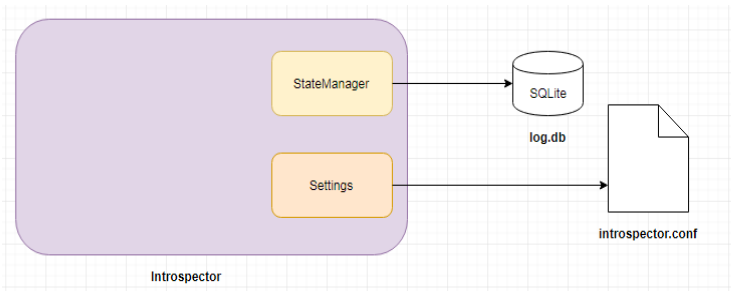

Figure 3: Introspector setup with Database

Next, a thread is spawned that opens a socket connection to listen for incoming requests for virtual machine administration from the Test Control Center while the main Introspector thread waits on another socket for incoming introspection requests. Administration requests are XML requests that have the root node <LibvirtServerMessage>

4. Research on Kernel Data Structures

The analysis conducted on the footprint of a process in kernel space is in near real-time, to distinguish between benign and malicious processes. The Linux Kernel maintains data structures that contain information about every action and resources used by a process. The study thus far has identified 15 features from 118 features which can effectively distinguish between benign and malicious processes [9] [10]. The next steps in the pipeline is to perform virtualization to extract threads, system call , invariant data structures like system call table and interrupt descriptor table, IP addresses, network sockets, URLs, open files, passwords, catches, clipboards and other user generated content, encrypted keys, and configurations of hardware and software [11].

5. Benign and Malware Applications

We have also performed research to extract the kernel data structures from the Linux kernel task structure using LibVMI library. The following Benign processes and Malwares samples have been identified to be installed on the Linux virtual machines. The Linux kernel version under consideration as of the writing of this paper is 3.16.0-23- generic.ko in Ubuntu 14.04 [11] [12].

GUI Applications	CLI Applications
1) Firefox - Web Browser 2) Thunderbird - Email client application 3) gpaint - Paint Application 4) Libre - Office writer 5) Rythmbox - Music player 6) Connectagram - Word unscrambling game 7) Arora - Cross platform web browser 8) Empathy- Internet messaging application 9) Alarm clock 10) Shotwell- Photo manager	1) Alpine-Pico- Is a text editor, uses the pine email client for writing email messages. Run from the terminal as pico, pico.alpine 2) Aaphoto- An image manipulation tool for automatic color correction of photos 3) ACL2- Programming language in which user can model computer systems and a tool to help prove properties of those models. 4) Python- Programming language 5) Gedit Text editing tool 6) Calcoo - scientific calculator 7) Calcurse- text based calendar and todo manager 8) Clamav-daemon- antivirus utility for unix-scanner daemon run from the terminal 9) Gzip- Zipping application

6. Linux Kernel Data Structure Extraction

At the outset we are going to extract the following task structure list from Linux data structures. Based on our research, the following list of features has been identified for extraction to test the virtualization framework in the pilot stage. These are the primary features of the processes that will run in the kernel of the Linux virtual machine at runtime [13] [14].

Sr. No.	Features' Name	Description
1	map_count	Number of memory regions of a process
2	page table lock	Used to manage the page table entries
3	hiwater rss	Number of page frames that a process owns
4	shared_vm	Number of pages in shared file memory mapping of a process
5	exec_vm	Number of pages in exec. memory mapping of process
6	nr_ptes	Number of page tables owned by a process
7	utime	Execution time of a process in user mode (tick count)
8	stime	Execution time of a process in kernel mode (tick count)
9	nvcsw	Volunteer context switches of a process
10	nivcsw	Involuntary context switches
11	total_vm	Size of process's address space in terms of Number of pages
12	min_flt	Minor page faults of a process
13	alloc lock.raw lock.slock	Used to lock memory manager, files and file system etc.
14	hiwater_vm	Max Number of pages appeared in memory region of process
15	fs.count	fs_struct's usage count to indicate the restrictions

7. Data Analytics and Results

The research on data analytics is currently on-going [15] [16]. The Data Analytics platform consists of centralized database servers for analytics and processing. The research will be focused on identifying the classification and anomaly detection machine learning algorithms that includes open source and commercial platforms, libraries with these platforms [17] [18] to detect malware [19] [20] that primarily focus on the Linux data structure extracted by the VMI Framework.

Once the framework is narrowed down and tuned as a viable solution for malware detection, the results will be summarized to demonstrate which machine learning models are powerful for malware detection and monitoring. The research outcome will consist of three components: memory data structures, algorithms with pros/cons and machine learning model performance data.

8. Conclusion and Future Work

The goal of the VMI framework research is to provide solid foundation for security of the Linux operating system as well as the capabilities to identify and monitor cyber threats in virtual environment. The key challenges of this research is to identify various data structures affected by modern malware, and in-depth examination of various machine learning algorithms with memory forensics to solve key cybersecurity issues.

The focus of this research is to develop a system that can work with the latest Linux operating systems providing memory forensics capability in a virtual environment. In the future, we plan to explore all possible ways of data analytics with big data technologies and deep learning. With sufficient time and effort, the development of this framework can result in an extremely powerful system for early threat detection and defense.

Acknowledgment

This research is supported by Department of Defense (DOD) – Test Resource Management Center (TRMC), USA. We thank our colleagues who provided insight and expertise, directly or indirectly, that greatly assisted the research.

References

[1] M H Ligh, A Case, J Levy, A Walters. "The Art of Memory Forensics", 2014
[2] M Wade, "Memory Forensics: Where to Start" at http://www.forensicmag.com/ article/2011/06/ memory-forensics-where-start, 2011
[3] Baliga, A., Ganapathy, V. and Iftode, L., Detecting kernel-level rootkits using data structure invariants. IEEE Transactions on Dependable and Secure Computing, 8(5), pp.670-684, 2011.
[4] D. Levy, H. A. Gohel, H. Upadhyay, A. Pons and L. E. Lagos, "Design of Virtualization Framework to Detect Cyber Threats in Linux Environment," 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, pp. 316-320, 2017
[5] H. Gohel. "Introduction to Network & Cyber Security", 2015
[6] Xen Project available at https://www.xenproject.org/, 2013
[7] H. Gohel. "Looking Back at the Evolution of the Internet." CSI Communications - Knowledge Digest for IT Community Vol. 38 Issue. 6 pp. 23-26, 2014
[8] N Joshi, D.B.Choksi, "Implementation of process forensic for system calls", International Journal of Advanced Research in Engineering and Technology (IJARET), ISSN 0976 – 6480(Print), ISSN 0976 – 6499(Online) Vol. 5, Issue. 6, pp. 77-82, 2014
[9] Blackbag Team, "MEMORY FORENSICS", at https://www.blackbagtech.com/blog/2016/03/07/ windows-memory-forensics/2016
[10] Farrukh Shahzad, M. Shahzad, Muddassar Farooq, "In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS" Information Sciences, Volume 231, pp. 45-63, ISSN 0020-0255,2013
[11] N Joshi, D. B. Choksi, "Implementation of Process Forensic for System Calls", International Journal of Advanced Research in Engineering & Technology (IJARET), Vol. 5, Issue. 6, pp. 77 - 82, ISSN Print: 0976-6480, ISSN Online: 0976-6499, 2014.
[12] H. Upadhyay, H. Gohel "Security Corner: Cyber Threat Analysis with Memory Forensics", CSIC – Knowledge Digest for IT Community, Vol. 40, Issue. 11, ISSN 0970-647X, pp. 17-19, 2017
[13] H. Upadhyay, H. Gohel "Design of Advanced Cyber Threat Analysis Framework for Memory Forensics", International Journal of Innovative Research in Computer and Communication Engineering, ISSN (Online): 2320-9801, ISSN (Print): 2320-9798, Vol. 5, Special Issue 2. pp.132-137, 2017
[14] F. Shahzad, S. Bhatti, M. Shahzad and M. Farooq, "In-Execution Malware Detection Using Task Structures of Linux Processes," 2011 IEEE International Conference on Communications (ICC), Kyoto, pp. 1-6, 2011.
[15] Z. Gu, Z. Deng, D. Xu and X. Jiang, "Process Implanting: A New Active Introspection Framework for Virtualization," IEEE 30th International Symposium on Reliable Distributed Systems, Madrid, pp. 147-156, 2011
[16] Hal Pomeranz, "Detecting Malware with Memory Forensics", at http://www.deer-run.com/~hal/ Detect_Malware_w_Memory_Forensics.pdf, 2015
[17] Hizver, Jennia, and Tzi-cker Chiueh. "Real-time deep virtual machine introspection and its applications." ACM SIGPLAN Notices. Vol. 49. No. 7. ACM, 2014.
[18] Hardik, Gohel. "Data Science - Data, Tools & Technologies." CSI Communications Knowledge Digest for IT Community, Vol. 39 Issue. 3, pp. 8-10, 2015
[19] H Gohel, P Sharma. "Study of Quantum Computing with Significance of Machine Learning." CSI Communications - Knowledge Digest for IT Community, Vol. 38, Issue. 11, pp. 21-23, 2015
[20] E Mariconti, O Lucky, P. Andriotis, "Detecting Android Malware by Building Markov Chains of Behavioural Models", NDDS'17, San Diego, USA, 2017