Overview of Virtual machine introspection technique and its usage in malicious image detection

Mohammad-Reza Memarian

05 Oct 2014

1 Introduction

A vulnerability in an image can infect all the users using from that image. Unlike conventional attacks on single systems, In this scenario attacker's code does not depend on the software stack of each individual user for exploitation. As Cloud handles high amount of data, attacker's pay off is highly large. One solution is to combine methods to detect Malicious Images and possible malware running on them. Virtual machine introspection (VMI), is one of he methods which is used in development of several forensic system.

2 Preliminaries

For having a better view and understanding of the VMI and use of it in the problem solving as well as discussed matters in this material, some preliminaries are discussed as follows.

2.1 Xen

Xen is a type 1 (baremetal) open source VMM (Virtual machine monitor) as there is no OS on the physical host. Xen does not provide a management interface so a special VM runs on the system at all times. In Xen, VMs are refereed to as domains. The one VM which is privileged in called Dom0 and the others are called DomU (Dom1, Dom2,..., DomN). VMM gives Dom0 system access to a control library which lets the system administrates to create, destroy, start, pause, stop and allocate resources to VMs from Dom0 machine. Dom0 can also request that memory pages allocated to unprivileged VMs be available to the Dom0 system. This matter makes it possible for the VMI application to run within Dom0 to view memory of other VMs. So this privilege should only be given to Dom0 VM and other VMs should only be restricted to have access to the memory space that is allocated to them by VMM.

2.2 VMI

According to [1], VMI is a technique that allows one guest VM to monitor, analyze and modify state of another VM by observing its virtual memory page. VMI is used by security related software to inspect VMs. There are drawbacks as well. Malwares can detect they are monitored by other virtual machines by use of side-channel attack as a result of shared physical resources technology. VMI techniques are used in digital forensic, development of security related software like IDS and other commercial products. According to [2], VMI tools inspect a VM from outside to assess what's happening on the inside. This makes it possible for security tools such as virus scanners and intrusion detection systems to observe and respond to VM events from a safe location outside a monitored machine. In this case malicious code can not attack to local monitoring systems as they are observing from outside of monitored VM. From process perspective, processes request to access to memory addresses directly (with the assumption of OS abstraction in middle). Operating system although abstracted from process but plays an active role in providing memory location address. But in the case of virtualization, Virtual machine monitor (VMM) provides an abstraction layer between each VM OS's memory management and underlying physical hardware. Because of VMM's active involvement in this process and its elevated privileges, it can also access each VM's memory page directly without VM requesting the page. As a result VMM can make those pages available to other VMs. This routine facilitate VMI process. There are various systems that have implemented VMIs such as [3] and [4]. In traditional digital forensic, snapshot of a system had to be analyzed so important data and evidence on existed on memory would be lost. by using VMI, just by reading the VM memory, the system state can be in hand.

2.2.1 VMI categorization

VMI systems drop in one of two categories, the ones that monitor the subject behavior and the ones that interfere with subject behavior. So security mechanisms using VMI to monitor a system such as livewire can only detect and report problems, but ones that can interfere like lycosID and uDenali can respond to a detected threat such as terminate the relevant process in the VM or reduce the resources available to the VM. So the method of event reply determines whether analysis must be performed in real-time as the target system executes or at some later time's under analyst's control.

2.2.2 VMI Implementation

VMI application can be implemented in atleast one of the following two locations. First, to embed the VMI application in the VMM itself which requires VMM code to be modified and makes VMI application highly dependent on VMM version. second, to put VMI outside of the VMM. For example in case of Xen, VMI can be placed in Dom0 VM which is privileged VM. So because tool

interact through a standard API, it is less likely pron to change in case of VMM changes. But it may reduce the VMI application ability to do inline processing.

2.3 LibVmi

LibVMI is an introspection library that is used for reading from and writing to VM's memory. LibVMI also provides functions for accessing CPU registers, pausing and unpausing a VM and printing binary data. LibVMI currently supports VMs running in either Xen or KVM virtualization platforms. LibVMI also supports reading physical memory snapshots when saved as a file.

2.4 MiniOs

MiniOs is an extremely small OS that originally began as an example from the Xen community of how to port a kernel to the Xen architecture for para-virtualizaed application. MiniOs does not have user space and multi-threating capabilities.

3 Related works

In [1], a paper which is published in IEEE cloud engineering conference in 2014, a very light solution for detection of malicious images using VMI has been proposed. They used from MiniOS, a virtualized operating system that uses minimal resources on Xen virtualization platform. Using that virtualized OS, memory space of other virtual machines are analyzed. These detector virtual machines are called Forensic Virtual Machines (FVMs). Due to light nature of FVMs, numerous instances can run to find symptoms in an environment. Basically FVMs look for unusual and malicious symptoms not the malicious behavior. Each FVM is responsible for detection of one symptom. These symptoms are detected based on the strategy that malware writers reuse from components of each other so the re-usage of components produce symptoms among even malware families. These symptoms can be as simple as changing a registry key in windows systems in most of the malwares or stopping processes such as anti virus. So FVMs are small and lightweight VMs that using VMI, discover symptoms in real time. FVMs exchange detected information using a secure channel. When a symptom is detected, it is reported to Command and Control via Domain0 (Dom0). Command and control can help to freeze the VMs by denying any CPU cycle. So Command and control correlates all the info to get the appropriate decision. FVMs include a set of distribute algorithms that determines next VM which should be analyzed plus the determined time. In previous sections we described that VMI should either be implemented in VMM or a another guest VM. Also we described in Xen, Dom0 is the privileged VM. But in [1], MiniOs used in a guest OS which is not Dom0. As a result Xen source code has been modified to allow access to FVMs. But as stated in the paper too, there has not been introduced a way to distinguish VMs from FVMs. As

a result, research in an access control mechanism to handle access management of FVMs to memory space of observed VMs are needed.

In [5], a paper published in CCS 2014, Although the focus is not on finding malware using VMI, but an encrypted VMI system has been designed. The problem with implementing VMI in the public clouds is that VMI can release information about user's application status. So there will be possibility of exposing confidential information from VMs. This paper proposed CryptVMI, an encrypted virtual machine introspection system that provides users complete status of their VMs while keeping confidentiality from possible attackers including administrators.

References

[1] J. S. K. H. C. I. D. Adrian L. Shaw, Behzad Bordbar, "Forensic virtual machines: Dynamic defence in the cloud via introspection," in *IC2E 2014: IEEE International Conference on Cloud Engineering.*

[2] B. H. Kara Nance, "Virtual machine introspection: Observation or interference?" University of Alaska, Tech. Rep., 2008.

[3] A. C. A.-D. Stephen T. Jones and R. H. Arpaci-Dusseau, "Vmm-based hidden process detection and identification using lycosid," in *VEE 2008 Proceedings of the Fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments.*

[4] D. L. Lionel Litty, "Manitou: A layerbelow approach to fighting malware," in *ASID 2006 Proceedings of 1st workshop on Architectural and system support for improving software dependability.*

[5] R. H. C. Fangzhou Yao, Read Sprabery, "Cryptvmi: A flexible and encrypted virtual machine introspection system in the cloud," in *SCC 2014 Proceeding of 2nd international workshop on Security in cloud computing.*