Guest-Based System Call Introspection with Extended Berkeley Packet Filter

by

Huzaifa Patel
Supervisor: Anil Somayaji

A thesis pre-proposal submitted to the School of Computer Science in partial fulfillment of the requirements for COMP 4906

Carleton University
Ottawa, Ontario
September 2022

Main Objectives

Framework Implementation

Our goal is to implement a virtual machine introspection (VMI) system in which we monitor virtual machine (VM) system call sequences from the hypervisor using Extended Berkeley Packet Filter (eBPF). While monitoring system calls, our goal is to detect anomalous VM system calls and respond to them by slowing down the process responsible for the anomalous system call(s). The VMI system will rely on both a host running Intel x86 architecture, and a Kernel Virtual Machine (KVM) hypervisor that uses the Intel Virtualization (VT-x) extension.

Measuring our Frameworks Performance

It is important that our implemented framework is efficient enough to run on computer systems. For this reason, performance analysis will be carried out to determine the efficiency of our framework and its effects on the KVM hypervisor and its VMs. We intend to do tests on different hardware, but on the same Linux Kernel version to ensure consistency within the KVM guest.

Research on Related Work

We are also interested in conducting research on work similar to our own. For example, there exists a framework named "Nitro" [1] that makes use of VMI to monitor guest system call traces. Nitro is extremely flexible as it has been proven to work on Windows, Linux, 32-bit, and 64-bit environments [1]. It also surpasses other similar VMI systems in both performance and functionality [1]. Unlike our proposed framework, the current version of Nitro is not able to detect and react to malicious system calls.

Research on KVM Hypervisor Security

As our framework derives abstracted VM system call information from the KVM hypervisor, we rely on the integrity of the data given to us by the KVM hypervisor.

This means that the security of our framework relies on the security of the KVM hypervisor. Hence, it is critical to validate and prove that the information we are recieving from the KVM hypervisor is not being altered. For this reason, we will do this by conducting research on the security of the KVM hypervisor.

Motivation

Implementing an out-of-VM VMI framework to monitor, detect, and respond to guest system call sequences has many advantages over traditional in-VM monitors. Firstly, an out-of-VM VMI system can run at a higher privilege level, allowing it to have a view of system calls of every VM. Secondly, an out-of-VM VMI system is isolated from attacks that originate from within the guest OS, and is not directly visible from the guest. Lastly, as the VMI system is one layer below the guest OS, it is possible to interfere with VM processes that are discovered to be making malicious system calls.

Instead of system call sequences, a neural network implementation is a modern approach to solving the problem of detecting malicious processes. Although system call sequences requires a less complex implementation than that of a neural network implementation, we believe complexity does not equate to better. Our motivation for using an implementation based on system call sequences is because we believe it is still effective in detecting malicious processes.

Bi-weekly Schedule

Fall 2022	**Proposed Commitement**
Week 1	Begin research on the background and related work.
Week 3	Write about the background, related work, and the problem that is being solved,
Week 5	Write about our implementation plans and how we plan on improving upon related work.
Week 7	Begin implementation of our VMI framework.
Week 9	Continue implementing our VMI framework.
Week 11	Write about our implementation.
Week 13	Finalize thesis proposal, and write up our plan of action for the second term.

References

[1] Pfoh J., Schneider C., Eckert C. Nitro: Hardware-based System Call Tracing for Virtual Machines. https://www.sec.in.tum.de/i20/publications/nitro-hardware-based-system-call-tracing-for-virtual-machines. (2011).