Guest-Based System Call Introspection with Extended Berkeley Packet Filter

by

Huzaifa Patel

A thesis proposal submitted to the School of Computer Science in partial fulfillment of the requirements for the degree of

Bachelor of Computer Science

Under the supervision of Dr. Anil Somayaji

Carleton University

Ottawa, Ontario

September, 2022

their kindness is masquerade.
yearning to occupy ones with false pretenses.
it's used to sedate.

Abstract

Acknowledgments

I want to express my heartfelt gratitude to my supervisor, Dr. Anil Somayaji for providing me with the opportunity to work on a thesis during the final year of my undergraduate degree. Unlike previous variations of the Computer Science undergraduate degree requirements, completing a thesis is no longer a prerequisite. Therefore, I prostualte it is a great privlidge and honor to be given the opportunity to enroll into a thesis-based course during ones undergraduate studies.

I did not have prior experience in formal research when I first approached Dr. Somayaji. Despite this shortcoming, it did not stop him from investing his time and resources towards my academic growth. Without his feedback and ideas on my framework implementation and writing of this thesis, as well as his expertise in eBPF, Hypervisors, and Unix based Operating Systems, this thesis would not have been possible.

I would like to commend PhD student Manuel Andreas from The Technical University of Munich for introducing me to the concept of a Hypervisor. Without him, I would not have approached Dr. Somayaji with the intention of wanting to conduct research on them. His minor action of introducing me to hypervisors had the significant effect of inspiring me to write a thesis on the subject. I also want to thank him for his willingness to endlessly and tirelessly teach, discuss and help me understand the intricacies of hypervisors, the Linux kernel, and the C programming language.

I would also like to thank Carleton University's faculty of Computer Science for their efforts in imparting knowledge that has enthraled and inspired me to learn all that I can about Computer Science.

I would like to extend my appreciation to the various internet communities which have provided the world with invaluable compiled resources on hypervisors, Unix based operating systems, eBPF, the Linux kernel, the C programming language, and Latex, which has helped me tremendously in writing this thesis.

Finally, I would like to thank my family for their encouragement and support towards my research interests and educational pursuits.

Contents

Nomenclature

KVM	Kernel Virtual Machine
OS	Operating System
VMI	Virtual Machine Introspection
CPU	Central Processing Unit

Introduction

On desktop computers, it used to be neccessary to close one application in order to open another. With advancements in computing, it became possible for users on one computer and one operating system to execute numerous applications at once. Users can now switch between one machine, many operating systems, and multiple applications at their discretion thanks to virtualization.

Virtualization is a technology that makes it possible for multiple operating systems (OSs) to run concurrently, and in an isolated environment on a single physical machine. Virtualization is the use of a computer's physical central processing unit (CPU) to support the software that creates and manages virtual machines (VMs). The software that creates and manages virtual machines is formally known as a hypervisor. The operating system, when running a hypervisor, is called the host, while the virtual machine that use its resources is known as the guest.

There are many techniques to achieve virtualization of a computer system.

Hardware-assisted

There are 12 projects that use the guest-assisted approach. The pioneer work, LARES [Payne et al. 2008], inserts hooks in a guest VM and protects its guest component by using the hypervisor for memory isolation with the goal of supporting active monitoring. Unlike passive monitoring, active monitoring requires the interposition of kernel events. As a result, it requires the monitoring code to be executed inside the guest OS, which is why it essentially leads to the solution of inserting certain hooks inside the guest VM. The hooks are used to trigger events that can notify the hypervisor or redirect execution to an external VM. More specifically, LARES design involves three

components: a guest component, a secure VM, and a hypervisor. The hypervisor helps to protect the guest VM component by memory isolation and acts as the communication component between the guest VM and the secure VM. The secure VM is used to analyze the events and take actions necessary to prevent attacks.

1.1 Motivation

1.1.1 Why Design a New Framework?

1.1.2 Why Out-Of-VM monitor?

1.1.3 Why eBPF?

1.1.4 Why Sequences of System Calls?

1.2 Problem

1.2.1 The Semantic Gap Problem

1.3 Approaching the Problem

1.4 Contributions

1.5 Thesis Organization

Background

2.1 Intel Virtualization Extention (VT-X)

2.2 The Kernel Virtual Machine Hypervisor

2.2.1 Model Specific Registers

2.2.2 VMCS

2.2.3 VM ENTRY Context Switch

2.2.4 VM EXIT Context Switch

2.3 QEMU

2.4 System Calls

2.5 Virtual Machine Introspection

2.6 eBPF

2.7 The Linux Kernel Tracepoint API

2.8 pH-based Sequences of System Call

Related work

3.1 Nitro: Hardware-Based System Call Tracing for Virtual Machines

Implementing Frail

4.1 User Space Component

4.2 Kernel Space Component

4.2.1 Custom Linux Kernel Tracepoint

4.2.2 Kernel Module

4.3 Tracing Processess

4.4 Proof of Tracability of all KVM Guest System Calls

Threat Model of Frail

Future Work

Conclusion

References