Trap or Trace: Implementing Kernel-space Data Watchpoints Efficiently

Peter Goodman Angela Demke Brown
Department of Computer Science
University of Toronto
{pag, demke}@cs.toronto.edu

Akshay Kumar Ashvin Goel
Department of Electrical and Computer Engineering
University of Toronto
{akshay, ashvin}@eecg.toronto.edu

Abstract

Finding, understanding, and fixing bugs in operating system kernels is challenging. Watchpoints (also called data breakpoints) are a powerful tool that help with this problem; however, hardware-based watchpoints are too scarce for many uses and existing software-based watchpoints are either slow, or lack contextual information about the watched memory.

In this paper, we introduce *behavioral watchpoints*: a new form of software-based watchpoints. Behavioral watchpoints extend on traditional watchpoints in novel ways: they watch ranges of addresses and they maintain context-specific information about watched memory, enabling debugging applications that cannot be easily implemented using traditional hardware or software watchpoints. For example, our framework can specialize instrumentation based on the context in which a watchpoint is triggered, which makes it possible to limit the scope of heavy-weight instrumentation to only code operating on watched data.

We discuss and evaluate several implementation strategies for our behavioral watchpoints framework, ranging from trap-based watchpoints (data-centric) to whole-kernel instrumentation based watchpoints (code-centric). We show that behavioral watchpoints, as implemented within a kernel dynamic binary translation framework, have reasonable overheads for their intended use in analysing and debugging operating system kernels.