Memory Safety and Untrusted Extensions for TinyOS

John Regehr Nathan Cooprider
Will Archer Eric Eide

UUCS-06-007

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

June 30, 2006

Abstract

Sensor network applications should be reliable. However, TinyOS, the dominant sensor net OS, lacks basic building blocks for reliable software systems: memory protection, isolation, and safe termination. These features are typically found in general-purpose operating systems but are believed to be too expensive for tiny embedded systems with a few kilobytes of RAM. We dispel this notion and show that CCured, a safe dialect of C, can be leveraged to provide memory safety for largely unmodified TinyOS applications. We build upon safety to implement two very different environments for TinyOS applications. The first, Safe TinyOS, provides a minimal kernel for safely executing trusted applications. Safe execution traps and identifies bugs that would otherwise have silently corrupted RAM. The second environment, UTOS, implements a user-kernel boundary that supports isolation and safe termination of untrusted code. Existing TinyOS components can often be ported to UTOS with little effort. To create our environments, we substantially augmented the CCured toolchain to emit code that is safe under interrupt-driven concurrency, to reduce storage requirements by compressing error messages, to refactor direct hardware access into calls to trusted helper functions, and to make safe programs more efficient using whole-program optimization. A surprising result of our work is that a safe, optimized TinyOS program can be faster than the original unsafe, unoptimized application.