Poster: Memory Protection in Ultra-Low-Power Multi-Application Wearables

Taylor Hardin†, Josiah Hester*, Patrick Proctor†, Jacob Sorber*, David Kotz†
†Dartmouth College and *Clemson University

1. INTRODUCTION

An increasing number of wearable devices support the execution of multiple third-party applications, increasing the functionality and flexibility of these devices. These multi-application, multi-tenant devices provide users with more options, and application developers with a standard platform. Typical ultra-low-power wearable devices, however, lack the type of hardware memory protection mechanisms – such as Memory Management Units (MMU) – needed to safely separate applications. At best, they provide a Memory Protection Unit (MPU), which allows the user to configure read/write/execute permissions for a few distinct regions of memory. At worst, no hardware memory protection is provided. MPU capabilities vary across hardware platforms, with many shortcomings: (1) the MPU may only support a few distinct memory regions (fewer than one per application), (2) the MPU may not protect all regions of memory, like hardware registers, and (3) MPU protection boundary rules can be arcane, because they depend on opaque hardware implementations. Our key observation is that by supplementing a limited segment MPU with runtime checks, and using compile-time static analysis to explicitly layout applications in memory, we can guarantee application isolation (sandboxing) even on these limited MPUs, with lower overhead than software-only solutions.

2. APPROACH

Ultra-low-power microcontrollers have historically not offered MPUs; only recently have MPUs become more prevalent, but many lack the functionality for sufficient memory management and protection. Thus, those who develop multi-application, multi-tenant platforms isolate applications using compile-time or run-time software sandboxing (e.g., AmuletOS [1]), imposing limits on application developers and adding time/space overhead to running applications. We have developed methods, however, to leverage the limited MPUs and thereby reduce overhead cost by narrowing the use of software-based approaches.

MobiSys'17 June 19-23, 2017, Niagara Falls, NY, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4928-4/17/06.
DOI: http://dx.doi.org/10.1145/3081333.3089314

Our prototype solution runs on the MSP430 microcontroller (specifically, MSP430FR5989); its MPU supports four memory regions (with some restrictions). At compile time, our solution splits memory into three coarse-grained sections: 1) OS stack, 2) OS code, and 3) applications. Each application in Section 3 is then further divided into code and stack segments. By providing each application with its own stack we can ensure that applications are not able to glean or modify sensitive stack information from the system or other applications. This layout allows us to use the MPU to prevent all illegal accesses above the currently executing application's stack. Since we need only check memory references below the current application's stack, our method eliminates half of the number of compiler-inserted runtime checks. The compiler inserts code for MPU management, enabling cross-domain "system calls" from the application to the OS, and arranging the MPU boundaries when the OS or another application needs to run next.

3. CONTRIBUTION

The poster summarizes the current state of MPU hardware used in ultra-low-power wearables, and evaluates our prototype solution. For applications with a high frequency of memory accesses, our approach induces less overhead than software-only solutions, and gives more freedom to application developers (by avoiding the need for language or compile-time restrictions imposed by many software solutions). With minimal changes, our solution can be applied to other MPUs with varying degrees of functionality. Given these observations, we propose changes to existing MPU hardware that would decrease runtime overhead and/or eliminate the need for software checks. Please follow us at amulet-project.org.

Acknowledgements. This research results from a research program at the Institute for Security, Technology, and Society, supported by the National Science Foundation under award numbers CNS-1314281, CNS-1314342, CNS-1619970, and CNS-1619950. The Amulet logo was designed by the DALI Lab at Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

4. REFERENCES

[1] J. Hester et al. Amulet: An Energy-Efficient, Multi-Application Wearable Platform. In *Proceedings of the ACM Conferences on Embedded Networked Sensor Systems (SenSys)*, pages 216-229. ACM, Nov. 2016.