IWES 2020
5TH ITALIAN WORKSHOP ON EMBEDDED SYSTEMS
8/9 FEBRUARY 2020

Enforcing Control-flow integrity in Virtualized environments on ARM platforms

Gabriele Serra, Pietro Fara, Giorgiomaria Cicero, Alessandro Biondi
Scuola Superiore Sant'Anna
name.surname@santannapisa.it

Abstract

Virtualization is becoming a key technology for embedded systems designs, especially for applications with mixed-criticality and security levels. Consequently, safety-critical OSes more susceptible to the most common malicious cyber-attacks such as code-reuse attack (CRA) or return-oriented programming (ROP). The control-flow integrity (CFI) technique is one of the most efficient to counteract this kind of attacks. CFI is undoubtedly a powerful technique but scarcely applicable in real cases, especially for the overhead introduced to ensure complete graph enforcement. To make some CFI techniques implementable in practical applications, manufacturers have started to offer hardware supports. Our work focuses on exploiting the hardware mechanisms offered by ARM processors called *pointer authentication* and *branch-target identification* to realize a robust CFI enforcement providing a hypervisor-centric attack-detection and recovery strategy. Improvement of keys management can be realized with little effort exploiting the capabilities of the virtualization extension offered by ARM, such as the TrustZone. Conversely, PA attack-detection is challenging to recognize due to architectural limitations; the forthcoming introduction of FPAC extension in version 8.6-A of the ARM architecture solves the problem at the root, allowing proper attack detection/recovery strategies. Furthermore, the lack of support of security mechanisms for legacy architecture is an issue when dealing with portable software. We counteracted all these issues taking advantage of a type-1 hypervisor named Clare developed at our laboratory. Furthermore, we realized an emulation of the PA mechanism through both a full-software approach and a hybrid software-hardware approach employing an FPGA. Firsts results show an overhead limited and less than 10%. Our current investigations focus on improving the protection model to reduce the total overhead introduced by the mechanisms, for instance, protecting only vulnerable sections of code and limiting the usage to a restricted set of functions.