HardBlare: an efficient hardware-assisted DIFC for non-modified embedded processors

Pascal Cotret a, Guillaume Hiet b, Guy Gogniat c, Vianney Lapôtre c

a SCEE/IETR, CentraleSupélec, Cesson-Sévigné - FRANCE
b CIDRE/INRIA, CentraleSupélec, Cesson-Sévigné - FRANCE
c Lab-STICC, University of South Brittany, Lorient - FRANCE

Introduction

Classic IT security context: main goals are security policy definition (in terms of confidentiality/integrity), its enforcement (malicious content filtering, isolation...) and its monitoring (intrusion detection, vulnerability analysis).

DIFC (Dynamic Information Flow Control) is an approach aiming to ensure that security properties are preserved all along the execution and files storage.

Containers of information: files, variables... Labels (also known as tags) are attached to such containers. Information flow policy ⇒ relation between tags.

Existing Hardware-Assisted Approaches

- In-core DIFT
  - Minimal impact on speed
  - Tag propagation in core pipeline + not portable
- Offloading DIFT
  - Separate processor for DIFC controls
  - Inter-core logic + power consumption × 2
- Off-core DIFT
  - Dedicated co-processor
  - Nearly no modification of the main processor
  - Main core may stall

HardBlare Approach

HardBlare targets heterogeneous SoCs combining a hardcore processor based on the ARMv7 architecture with a FPGA (Zynq from Xilinx, SoC from Altera).

Main processor must send a tuple composed of the following elements:
- Instruction pointer (i.e. program counter).
- Instruction encoding (fine-grained level).
- Memory addresses and processed registers (in case of load/store instructions).

As we cannot modify the ARM chip, how can we get the needed information from the main processor?
- A trace mechanism should help to perform DIFC controls.
- Processor/coprocessor synchronization:
  - Performed on system calls (a few hundred cycles).
  - Synchronization time overhead: around ten cycles.

⇒ Evaluation in terms of performances and security on a full-fledged Linux system with standard binaries.

Some References


Main Contributions at a Glance

- Hardware-assisted DIFC system with limited time overheads.
- Approach based on a non-modified CPU with a standard Linux and generic binaries. Could be implemented by industrial partners in medium-term.
- Hardened with hardware security mechanisms: trusted coprocessor storage and bus protection in terms of confidentiality/integrity.
- Contributions on software-related issues as well (static/dynamic IFC analysis, aka hybrid analysis).
- Perspectives on runtime reconfiguration and multicore manycore systems.

Software-Based DIFC

Analysis can be done at the operating system and application levels:
- OS-level: dedicated OS or modification of existing OS.
  ✓ Small overhead (< 10%).
  ✗ Overapproximation issue.
- Application-level: machine code or specific language.
  ✓ Gain in precision (hybrid analysis).
  ✗ Huge overhead (> × 3 at least!)

Goals

- Modify existing hardware to accelerate IFC and protect tags.
- Compatibility with existing software/hardware.
- Design of a flexible solution (no dependency with policy or tag format).