Provably Secure Execution Platforms for Embedded Systems

The PROSPER Project

Mads Dam
KTH Royal Institute of Technology

Programvara för konkurrenskraft, SSF, Vinnova, 10 Feb 2015
The Evolving Security Landscape

Increasing attack surfaces
Increasing aggregate value
Increasing attack sophistication
Increasing demand for strong security solutions
Industrial pull for strong verification techniques
Significant exploitation potential
It’s the Execution Platform, Stupid!

Processor hardware: a shared commodity
- User, payment provider, media owner, platform provider, operating system, ...
- All need private, tamperproof storage and cpu cycles
- Without this, security will remain fragile

Need trustworthy execution environments
- Allow secure sharing of hardware
- Memory isolation
- Minimal
- Open to scrutiny
Secure Virtualization

Applications

OS

Android

Secure OS

Hypervisor

CPU

ARM
PROSPER Objectives

Build the next-generation framework for fully verified, secure hypervisors for embedded systems

Demonstrate utility using
- Commodity hardware – ARMv7/v8, MIPS, ...
- Commodity software – Linux, RTOS, ...

Develop the required verification technology
- Theory
- Tools
- For security formalization and analysis
- For hypervisor development
Demonstrator: Provably Secure Kernel Updates in Linux on ARM

**Objective:** Execute only signed programs
What Does This Give Us?

**Theorem:** Assume:

1. The processor model is correct.
2. The hypervisor is securely initialized.
3. The initial Linux image is signed.

Then:

Only signed code is ever executed
What Does This Give Us?

**Theorem:** Assume:
1. The processor model is correct.
2. The hypervisor is securely initialized.
3. The initial Linux image is signed.

Then:
- Only signed code is ever executed, AND
- Hypervisor is never tampered with
What Does This Give Us?

**Theorem:** Assume:

1. The processor model is correct.
2. The hypervisor is securely initialized.
3. The initial Linux image is signed.

Then:

- Only signed code is ever executed, AND
- Hypervisor is never tampered with, AND
- Monitor is never tampered with
What Does This Give Us?

**Theorem:** Assume:
1. The processor model is correct.
2. The hypervisor is securely initialized.
3. The initial Linux image is signed.

Then:
- Only signed code is ever executed, AND
- Hypervisor is never tampered with, AND
- Monitor is never tampered with, AND
- No memory page is ever simultaneously write and execute enabled, ...
What Does This Give Us?

**Theorem:** Assume:
1. The processor model is correct.
2. The hypervisor is securely initialized.
3. The initial Linux image is signed.

Then:

Only signed code is ever executed

This is a big deal!
What Does This Give Us?

**Theorem:** Assume:
1. The processor model is correct.
2. The hypervisor is securely initialized.
3. The initial Linux image is signed.

Then:
   
   Only signed code is ever executed

   **This is a big deal!**

Theorem about security inside COTS OS
No Linux bug can cause unsigned code to be executed
Use to bootstrap many provably secure services
How Do We Do This?

Theorem proving + binary verification

- Formal processor ISA specification
  - Including memory management unit
  - Large task
- Top level security specification:
  Processor + idealized MMU behaviour
- Low level implementation
- Proof of equivalence

• Repeat as needed
PROSPER Results So Far

Hypervisor #1:
• Separation kernel for ARMv7
• Fully verified at assembly level
• Papers in CCS’13, CPP’13, TrustED’13

Hypervisor #2:
• Linux-on-ARM hypervisor
• Partially verified at assembly level
• Papers in CCS’14, Sofsem’15

Lots of auxiliary theory and tools
PROSPER Results To Come

Hypervisor #3:
- Linux-on-ARM hypervisor for 64 bit ARMv8
- Much more complex hardware:
  - Multicore essential
  - Virtualization support

To do:
- ISA-level multicore model + validation
- Processor architecture analysis tool
- Verification support for concurrent machine code
- Hypervisor design and analysis
Spin-offs and Collaborations

Vinnova UDI project HASPOC w. Ericsson, Sectra, T2Data, Tutus, Atsec, SICS, KTH
1 patent application
2 PhD’s completed
Open source hypervisor release
SICS project with Ericsson Research
EU FP7 project UaESMC, KTH
MONITOR project KTH-Ericsson
HOL4 Cambridge team (Fox, Myreen)
TU-Berlin and DT Labs (Seifert)
INRIA/IRISA/DGA Rennes

...
Team

At KTH:
• Mads Dam, Prof, PI
• Roberto Guanciale, postdoc
• Christoph Baumann, postdoc
• Hamed Nemati, PhD student
• MSc students
• (Andreas Lundblad, PhD)
• (Gurvan le Guernic, postdoc)
• (Narges Khakpour, postdoc)
• (Musard Balliu, PhD)

At SICS:
• Christian Gehrmann, PhD, co-PI
• Arash Vahidi, PhD
• Oliver Schwarz, PhD student
• Viktor Do, researcher
• MSc students
Main Publications

- Dam, Lundblad, le Guernic: “TreeDroid: a tree automaton based approach to enforcing data processing policies”, Proc. CCS’12
- Dam, Guanciale, Khakpour, Nemati, Schwarz, "Formal Verification of Information Flow Security for a Simple ARM-Based Separation Kernel", CCS’13
- Khakpour, Schwarz, Dam, "Machine Assisted Proof of ARMv7 Instruction Level Isolation Properties", Proc CPP’13
- Dam, Guanciale, Nemati, “Machine code verification of a tiny ARM hypervisor”, in Proc. TrustED’13
- Balliu, Dam, Guanciale: Automating information flow analysis of low level code, CCS’14
- Schwarz, Dam: Formal verification of secure user mode device execution with DMA, HVC’14
- Nemati, Guanciale, Dam: Trustworthy isolation of the ARMv7 memory subsystem, Sofsem’15