Can Truly Dependable Systems Be Affordable?

Gernot Heiser
NICTA and UNSW
Present Systems are NOT Trustworthy!

Yet they are expensive:
• $1,000 per line of code for “high-assurance” software!
Fundamental issue: large stacks, need isolation

E.g. medical implant

- Processor
  - RTOS
    - Device drivers (1,000 LOC)
    - Network stacks (>10,000 LOC)
  - Control, monitoring, maintenance (>10,000 LOC)
- Life-supporting (1,000 LOC)
- 1,000 LOC

- 1 kLOC critical code
- 20–100 kLOC trusted computing base (TCB)
- 100s of bugs
- dozens of exploits!
High Assurance *Bad* Practice

- TCB of millions of LOC
- Expect 1000s of bugs
- Expect 100s of vulnerabilities

Hacker’s delight!

Isolation?

Xen/VMware/KVM hypervisor

Processor

Sensitive/critical/trusted

Uncritical/untrusted

Huge TCB
High Assurance Best Practice

- Isolate
- Minimise the TCB
- Assure TCB by
  - testing
  - code inspection
  - bug-finding tools

Always incomplete!

Uncritical/untrusted
Sensitive/critical/trusted

Separation kernel

Minimal “trusted computing base” (TCB)
State of the Art: NICTA’s seL4 Microkernel

- Provable isolation!
- Provable assurance!

No place for bugs to hide!

Strong Isolation

Uncritical/untrusted
Sensitive/critical/trusted

Truly dependable TCB

seL4 microkernel

Processor

©2013 Gernot Heiser, NICTA

APSys'13 Keynote
NICTA’s seL4: Mathematical Proof of Isolation

Confidentiality

Availability

Integrity

Abstract Model

Functional correctness [SOSP’09]

Isolation properties [ITP’11, S&P’13]

Exclusions (at present):
- Initialisation
- Privileged state & caches
- Multicore
- Covert timing channels

Translation correctness [PLDI’13]

C Implementation

Timeliness [RTSS’11]

Binary code

©2013 Gernot Heiser, NICTA
NICTA’s seL4 Microkernel: Unique Assurance

- First and only operating-system with functional-correctness proof: operation is always according to specification
- Predecessor deployed on 2 billion devices
- First and only operating-system with proof of integrity and confidentiality enforcement – at the level of binary code!
- World’s fastest microkernel on ARM architecture
- First and only protected-mode operating-system with complete and sound timing analysis
seL4: Cost of Assurance

- Confidentiality
  - 4.5 py

- Availability
  - Abstract Model
    - Proof
    - 20.5 py
    - 4.5 years
  - C Implementation
    - Proof
    - 2 py, 1.5 years
    - Mostly for tools
  - Binary code
    - Proof
    - 2 py, 1 year
    - Mostly for tools

- Integrity
  - 1 py
  - 4 months
  - 0 py
  - By construction

- $400 per line of code!
Cost of Assurance

Industry Best Practice:
• “High assurance”: $1,000/LOC, no guarantees, *unoptimised*
• Low assurance: $100–200/LOC, 1–5 faults/kLOC, *optimised*

State of the Art – seL4:
  – $400/LOC, 0 faults/kLOC
• Estimate repeat would cost half
  – that’s about the development cost of the predecessor Pistachio!
• Aggressive optimisation [APSys’12]
  – much faster than traditional high-assurance kernels
  – as fast as best-performing low-assurance kernels
What Have We Learnt?

Formal verification *probably* didn’t produce a more *secure* kernel

- In reality, traditional separation kernels are *probably* secure

But:

- We now have certainty
- We did it *probably* at less cost

**Real achievement:**

- Cost-competitive at a scale where traditional approaches still work
- Foundation for scaling beyond: 2 × *cheaper*, 10 × *bigger*!

**How?**

- Combine theorem proving with
  - synthesis
  - domain–specific languages (DSLs)
Next Step: Full System Assurance

DARPA HACMS Program:
- Provable vehicle safety
- “Red Team” must not be able to divert vehicle

Boeing Unmanned Little Bird (AH-6) Deployment Vehicle

SMACCMcopter Research Vehicle
System Structure

Control Board
- Monitor
- Control
- CAN bus controller
- Verified RTOS

Mission Board
- C&C
- File system
- Device drivers
- Untrusted Linux kernel, image processing
- seL4 – verified microkernel

Key:
- Trusted
- Trusted, NICTA
- Untrusted

Hardware
- CAN Bus

Hardware
- Process-
- sor
- C&C
- Radio
- Network
- Control
- Micro-
- controller
- Camera

©2013 Gernot Heiser, NICTA

APSys’13 Keynote
Architecting System-Level Security/Safety

Architecture Specification

Requirements
(specific set of security/safety properties)

Component Model

Untrusted

Automatic Analysis
(Requirements fulfilled)

Automatic Generation
of Glue code

Component Implementations

Untrusted

trusted

Correctness

Formal proof

Synthesis

Glue Code Proof

secured

Functional correctness

Security

Verified Glue Code

Communication

Init

seL4 Kernel

Correctness

Formal proof

Synthesis

Glue Code Proof

Secured

Functional correctness

Security

Verified Glue Code

Communication

Init

seL4 Kernel
Synthesis: Device Drivers [SOSP’09]

Formal OS Interface Spec

Formal Device Spec

Formalise specs!

driver.c
Actually works! (On Linux & seL4)

IDE disk controller  W5100 Eth shield  Intel PRO/1000 Ethernet
UART controller  Asix AX88772 USB-to-Eth adapter  SD host controller
Synthesis: Device Drivers

In progress:
- Extract device spec from device design work-flow
- Manual optimisations
- Verified synthesis

Formal OS Interface Spec

Formal Device Spec

driver.c
Hardware Design Workflow

Informal specification → High-level model → Manual transformation → Register-transfer-level description → netlist

- Low-level description: registers, gates, wires.
- Cycle-accurate
- Precisely models internal device architecture and interfaces
- “Gold reference”

Too detailed (for now)
Hardware Design Workflow

- Informal specification
- High-level model
- Register-transfer-level description
- netlist

- Captures external behaviour
- Abstracts away structure and timing
- Abstracts away the low-level interface

bus_write(u32 addr, u32 val)
{
    ...
}

Use for now
File-system properties:
- Multiple, pre-defined abstraction levels
- Naturally modular
- Lots of “boring” code
  - (de-)serialisation
  - error handling
Case study: Flash file system

- Linux-compatible
- Fits between VFS and flash abstraction (UBI)
Vision: Trustworthy System

Untrusted VM

Untrusted Apps

Untrusted Linux

Verified critical application

Verified High-level runtime

Verified Resource Management

Verified microkernel

Processor

Verified File systems

Verified Network Stacks

Verified Device Drivers

Devices
Lessons Learnt So Far

Formal methods are expensive?
• Cost-effective for high assurance on small to moderate scale
• $200-400/LOC for 10kLOC

We think we can scale bigger and cheaper:
• Componentisation
  – verify components in isolation – enabled by seL4 guarantees
  – cost – performance tradeoff
• Synthesis
• Abstraction: DSLs, HLLs increase productivity

Big challenge: Proof composition

The next few years will be exciting!