2023 T3 Week 11
seL4 in the Real World &
seL4 Research at TS@UNSW
@GernotHeiser
Copyright Notice

These slides are distributed under the Creative Commons Attribution 4.0 International (CC BY 4.0) License

• You are free:
  • to share—to copy, distribute and transmit the work
  • to remix—to adapt the work

• under the following conditions:
  • Attribution: You must attribute the work (but not in any way that suggests that the author endorses you or your use of the work) as follows:
    “Courtesy of Gernot Heiser, UNSW Sydney”

The complete license text can be found at http://creativecommons.org/licenses/by/4.0/legalcode
Today’s Lecture

• seL4 in the real world
  • HACMS & incremental cyber-retrofit
  • Adption and seL4 Foundation

• seL4-related research at UNSW Trustworthy Systems
  • Usability 1: Microkit
  • Usability 2: Lions OS
  • Pancake: Verifying device drivers
  • Verifying the seL4CP
  • Secure multi-server OS
seL4 in the Real World
DARPA HACMS (2012–17)

Retrofit existing system!

Unmanned Little Bird (ULB)

Develop technology

Off-the-shelf Drone airframe

Autonomous trucks

GVR-Bot
ULB Architecture

- Mission Computer
  - Ground Station Link
  - GPS
  - Camera
  - Network
- Flight Computer
  - Sensors
  - Motors
Incremental Cyber Retrofit
Incremental Cyber Retrofit

Original Mission Computer

Mission Manager
Crypto
Camera
Ground Station Link
Local Network
Linux

Trusted
GS Lk
Mission Mgr
Crypto
GPS
Linux
VMM

Trusted
Camera
Linux
VMM

Trusted
Crypto
Mission Mgr
Comms
GPS
VMM

Comp9242 2023 T3 W11: seL4 Deployments & seL4 Research at TS
© Gernot Heiser 2019-23 – CC BY 4.0
Incremental Cyber Retrofit

[Klein et al, CACM, Oct’18]
We brought a hackable quadcopter with defenses built on our HACMS program to @defcon AerospaceVillage. As program manager @raymondrichards reports, many attempts to breakthrough were made but none were successful. Formal methods FTW!
HACMS Outcomes & Consequences

- Demonstrated real-world suitability of seL4 and formal methods
  - Reversal of bad vibes from over-promising and under-delivering
  - Major re-think in US defence
- Dis-proved “security must be designed in from the start”
  - Retrofit is possible (under the right circumstances!)
- Led to follow-on funding for seL4 and deployment in the field
  - DARPA CASE, Feb’16 – Dec’22
  - seL4 Summits, since Nov’18 (initially sponsored by DARPA)
  - seL4 Foundation, since April’20
  - TII (UAE), Dec’21 – ongoing
  - NCSC (UK), Jan’22 – ongoing
  - DARPA PROVERS, ~Q1’24–Q3’26
seL4 in Products
Qiyan Wang, Global VP Engineering, Digital Systems
Electric car maker NIO

“this OS, based on seL4, will in our mass-production cars next year”
Usability 1: Microkit
Issue: seL4 Objects are Low-Level

>50 kernel objects for trivial program!
Simple But Non-Trivial System
Microkernel: Assembly Language of OS

seL4 provides

- threads
- scheduling contexts
- pages
- endpoints
- notifications

Programmer wants

- Processes
- Sockets
- Files

Result:

- everyone builds their own
- proliferation of bad designs
- huge waste of effort

Doing it right requires good abstractions
abstractions introduce policy limit application space
Step 1: seL4 Microkit

Minimal base for IoT, cyberphysical, other embedded use

• Restrict to static architectures
  • i.e. components & communication channels defined at build time

• Ease development and deployment
  • SDK, integrate with build system of your choice

• Retain near-minimal trusted computing base (TCB)
  • TCB suitable for formal verification

• Retain seL4’s superior performance
Microkit Abstractions

- Simple, single-threaded event-driven

Protection Domain 1
- `init(...)`
- `notified(...)`
- `protected(...)`

Protection Domain 2
- `init(...)`
- `notified(...)`

Communication
- `notify(...)`

Protected Procedure Call

Memory Region

- Minimal abstractions
- Thin wrapper of seL4
- Encourage “correct” use of seL4 primitives
- Static architecture

May be a virtual machine
libmicrokit: Event-handler loop

1. for (;;) {
2.     if (have_reply) {
3.         tag = seL4_ReplyRecv(INPUT_CAP, reply_tag, &badge, REPLY_CAP);
4.     } else if (have_signal) {
5.         tag = seL4_NBSendRecv(signal, signal_msg, INPUT_CAP, &badge, REPLY_CAP);
6.         have_signal = false;
7.     } else {
8.         tag = seL4_Recv(INPUT_CAP, &badge, REPLY_CAP);
9.     }
10.    event_handle(badge, &have_reply, &reply_tag, &notified);
11. }

COMP9242 2023 T3 W11: seL4 Deployments & seL4 Research at TS
libmicrokit: Invoking user code

1. event_handle(badge, &have_reply, &reply_tag, &notified) {
2.     if ((have_reply) = badge >> 63) {
3.         reply_tag = protected(badge & 0x3f, tag);
4.     } else {
5.         unsigned int idx = 0;
6.         do {
7.             if (badge & 1) {
8.                 notified(idx);
9.             }
10.            badge >>= 1; idx++;
11.         } while (badge != 0);
12.     }
13. }
Microkit System Description File (SDF)

1. `<system>
2.   <memory_region name="uart" size="0x1000" phys_addr="0x9000000" />
3.   <memory_region name="buf" size="0x1000" />
4.   <protection_domain name="serial" priority="250">
5.     <irq irq="33" id="0" />
6.     <program_image path="serial_server.elf" />
7.     <map mr="uart" vaddr="0x4000000" perms="rw" cached="false" ... />
8.     <map mr="buf" vaddr="0x4001000" perms="rw" setvar_vaddr="input" />
9.   </protection_domain>
10.  <protection_domain name="main">
11.    <program_image path="main.elf" />
12.  </protection_domain>
13.  </channel>
14.   <end pd="serial" id="1" />
15.   <end pd="client" id="0" />
16.  </channel>
17. </system>`
Verifying the Microkit: libmicrokit

C code

seL4 spec + Microkit spec

C parser from seL4 verification

SIMPL

SimpleExport from seL4 binary verification

SydTV-GL

Push-button!

SMT solver

Control-flow graph

yes/no
Verifying the Microkit: System Initialisation

- SDF
  - Generate
  - Gene-rate
- Isabelle SDF
- CapDL
  - Generate
  - Gene-rate
  - Translation Validation
- Isabelle CapDL

Links to initialisation proofs!

seL4 caps
Microkit Verification in Context

Conditions apply

CapDL spec

f(){
   ...
}

Microkit SDF

Proof-generating translation

PD1.c  PD2.c  libmicrokit.c

Compiler/Linker

system.elf

init.o

sel4 spec

microkit spec

Push-button proof
Microkit Status (evolving quickly…)

• Officially adopted by seL4 Foundation in Sep’23
• Supports AArch64, RV64 (x64 in progress)
• Verification presently for initial version only, catching up
• Dynamic features prototype:
  • fault handlers
  • start/stop protection domains
  • re-initialise protection domains
  • empty protection domains (for late app loading)
Usability 2: Lions OS

Current research
Lions OS: Highly Modular OS on Microkit

![Diagram showing the structure of Lions OS with layers for OS, Networking, IP Stack, Copier, Mux, NIC Driver, File System, Microkernel, and Hardware.]
Lions OS: Aims

**Fast:**
Best-performing microkernel-based OS ever

**Secure:**
Most secure real-world OS ever

**Adaptable:**
Suitable for a wide range of cyberphysical / IoT / embedded systems
Lions OS: Principles

**Overarching principle: KISS**
“Keep it simple, stupid!”

- Least Privilege
- Strict separation of concerns
- Radical simplicity
- Use-case–specific policies
- Design for verification
Least Privilege: Device Drivers

Time-honoured security principle [Saltzer & Schroeder, 1975]

Driver does not need access to data region!
Strict Separation of Concerns: Networking

Each component has one and only one job!
Radical Simplicity™

Provide **exactly** the functionality needed, not more

Simple programming model:
- strictly sequential code (Microkit)
- event-based (Microkit)
- single-producer, single-consumer queues
- ...

Static **architecture**, mostly static resource management
Driver Programming Model

- Lock-free bounded queues
- Single producer, single consumer
- Similar to ring buffers used by NICs

Driver model:
- Single-threaded
- Event-driven
- Simple!

Mostly moves pointers between rings

Packets to send

Buffers to reuse

Mux

Control region (Tx part)

NIC Driver

Tx Metadata reg.

NIC

Data region

Tx

Packets to send

Buffers to reuse

Mux

Control region (Tx part)

NIC Driver

Tx Metadata reg.

NIC

Data region

Tx

Packets to send

Buffers to reuse

Mux

Control region (Tx part)

NIC Driver

Tx Metadata reg.

NIC

Data region

Tx

Packets to send

Buffers to reuse

Mux

Control region (Tx part)

NIC Driver

Tx Metadata reg.

NIC

Data region

Tx
Use-Case–Specific Policies

Traditional OS: achieve adaptability by universal policies

Lions-OS: Use-case diversity through policies that are:
- optimised for one specific use case
- simple, localised implementation
- easy to replace by swapping component
Networking System

Client can be a VM

IP stack is library – not in system’s TCB!

Tx Mux encapsulates traffic-shaping policy

Most components small & simple – verification possible?

Client can be a VM

Driver can be a VM

IP stack is library – not in system’s TCB!
Comparison to Linux (i.MX8)

**Linux:**
- NW driver: 4k lines
- NW system total: 1M lines

**seL4 design:**
- NW driver: 700 lines
- MUX: 400 lines
- Copier: 200 lines
- IP stack: much simpler, client library
- Shared NW system total: < 2,000 lines

**Written by second-year student!**

Performance?
Evaluation Setup

- External load generator
- Client echoes packets

- 2 context switches per packet
- 10 context switches per packet

- External load generator
- Client echoes packets

- 2 context switches per packet
- 10 context switches per packet
Achieved Performance: i.MX8

- Gigabit Ethernet
- single core

Graphs Courtesy Lucy Parker
Achieved Performance: i.MX8

- Gigabit Ethernet
- multicore

Simplicity wins!

Graphs Courtesy Lucy Parker
Design for Verification

Verification enabled by:
• modularity
• radical simplicity
Lions OS Status

• Bulk of funding secured (DARPA, NIO, …)
• Networking system mostly done (Lucy’s thesis)
• File system prototype (design not final)
• First release in Dec’23
  • Complete point-of-sale system
  • Network, storage, touch screen, card reader
  • Components in Rust, Python
• Looking at push-button verification
Other TS Research

Verifying Device Drivers
Remember: Verification Cost in Context

Assurance

Cost ($/SLOC)

L4 Pistachio $100–150

$400

Fast!

Revolution!

?  seL4

Green Hills INTEGRITY $1000

Slow!
Driver Dilemma

- High seL4 verification costs partially due to C language
- Better language would reduce cost
- Drivers are low-level, need C-like language
- Drivers are commodity, must be cheap!
- seL4 is one-off, justifies cost

Idea:
1. Simplify drivers
2. Design verification-friendly systems language: Pancake
3. Automate (part of) verification

Lions OS
- Well-defined semantics
- Memory-safe

Verified compiler
CakeML: Verified Implementation of ML

- Mature functional language
- Large and active ecosystem of developers and users
- Code generation from abstract specs
- Re-use framework for new systems language: Pancake

https://cakeml.org
Pancake: New Systems Language

**Approach:**
- Re-use lower part of CakeML compiler stack
- Get verified Pancake compiler quickly
- Retain mature framework/ecosystem
Pancake Performance

Lions OS setup with Pancake Muxes

Performance degradation for well-understood reasons
**Pancake: New Systems Language**

**Status:**
- “Usable” rump language
- still requires C code for HW access
- inefficient invocation – re-initialise run-time each time
- Verified compiler
- Limitations well understood and remedies in progress

---

**CakeML**

---

**Pancake**
seL4-Related Research in TS

Secure Multi-Server OS
Recap: Secure Operating Systems

**Secure OS**: [Jaeger: OS Security]
Access enforcement satisfies the *reference monitor* concept

Enforces *mandatory protection*:
- non-bypassable
- tamperproof
- verifiable
Secure, General-Purpose OS

Aim: General-purpose OS that provably enforces a security policy

Requires:
- mandatory policy enforcement
- policy diversity
- minimal TCB
- low-overhead enforcement
Real-World Use
Courtesy Boeing, DARPA
Thank you!

To the brave AOS students for their interest and dedication
To the world-class Trustworthy Systems team for making all possible

Please remember to do the myExperience survey
There’ll also be a more detailed one we’ll invite you to fill in

John Lions Honours Scholarship closes this week!
https://www.scholarships.unsw.edu.au/scholarships/id/1757/6077