seL4 Verification

Danil Tolstov

Group: M3302

Contents

Overview of seL4

Key Features of seL4 Microkernel

seL4 A minimalistic open source microkernel providing core OS services:

- Process and thread management
- Memory isolation
- Secure inter-process communication

Key Features:

- Strong isolation ensures programs run in “sandboxes,” preventing interference.
- Optimized for resource-constrained devices.
- Supports ARM, x86, RISC-V.

Applications in Critical Systems

Widely used in various systems:

1. Aviation (e.g., Boeing, DARPA projects)
2. Medical devices (e.g., pacemakers)
3. Cybersecurity (e.g., secure smartphones)
4. Autonomous vehicles
5. Military systems (e.g., DARPA HACMS)
6. IoC

Why seL4?

- Verified reliability: No crashes or vulnerabilities.
- Ideal for life-critical and high-security systems.

Benefits of Formal Verification

In a nushell, **verification** is a mathematical proof that the kernel works as intended in all possible cases

Why It Matters

- **Reliability**: No crashes, even in edge cases.
- **Security**: Eliminates exploitable bugs.
- **Certification**: Meets standards like Common Criteria, DO-178C.
- **Uniqueness**: seL4 is the first fully verified general-purpose microkernel.

CommonCriteria and **DO-178C** are both certifications/standarts for security of software

Development

Development of the kernel started in 2000s by NICTA (Australian National Investigation Center).

It took 9 years, so that kernel was completed in 2009, and was fully verified in 2014.

Currently the kernel is supported by Data61, led by Gernot Heiser.

Code

The kernel is written in **C** (~9000 lines of code), with critical parts in **assembly** (~600 lines of code), totalling with almost 10000 lines of code verified using **Isabelle/HOL**

License: GPLv2 for open source; commercial licenses available.

Source code can be found on GitHub

Fundamentals of Formal Verification

What is Formal Verification?

- **Definition**: A mathematical process to prove that a system (e.g., seL4) behaves correctly for all possible inputs and scenarios.
- **Goal**: Ensure the system is free of bugs and adheres to its specification.
- **Example**: Proving that seL4's memory isolation prevents unauthorized access.

Formal Verification vs. Traditional Testing

- **Traditional Testing**:
  - Runs the system on a limited set of inputs.
  - Detects bugs but cannot prove their absence.
  - Example: Running seL4 with sample programs to check stability.
- **Formal Verification**:
  - Mathematically proves correctness for **all** possible cases.
  - Guarantees no bugs in verified properties.
  - Example: Proving seL4's IPC is secure for any input.
- **Key Difference**: Testing is incomplete; verification is exhaustive.

The Verification Process of seL4

History of seL4 Verification (2009–Present)

- **Timeline**:
  - **2009**: First release of seL4 by NICTA.
  - **2014**: Completion of initial formal verification (functional correctness).
  - **2016–Present**: Ongoing verification of additional properties (e.g., security, real-time).
- **Milestones**:
  - Verified on ARM, x86, and RISC-V architectures.
  - Adopted in critical systems (e.g., DARPA HACMS, Boeing).
- **Current**: Continuous updates and open-source contributions via Data61/CSIRO.

Stages of seL4 Verification

- **Formal Specification**:
  - Define precise, mathematical description of seL4's behavior.
  - Example: Specify memory isolation rules.
- **Modeling at Source and Binary Levels**:
  - Model seL4's C code and compiled binary code.
  - Ensure both levels behave identically.
- **Proof of Correspondence**:
  - Prove that high-level specification matches low-level implementation.
  - Example: Verify that C code implements specified IPC correctly.

Tools and Effort

- **Tools Used**:
  - **Isabelle/HOL**: Primary tool for theorem proving and specification.
  - **HOL4**: Additional theorem proving for specific properties.
  - **SMT Solvers**: Automated checking of logical constraints.
- **Effort**:
  - Over 20 person-years for initial verification (2009–2014).
  - Thousands of lines of proof code in Isabelle/HOL.
  - Ongoing maintenance requires additional person-years.
- **Impact**: seL4's verification is a landmark in systems software.