seL4 as a CPU Driver in an OS for Real Computers

Roman Meier, Zikai Liu, Ben Fiedler, Timothy Roscoe
Systems Group, ETH Zurich

July 20, 2024

Modern operating systems – including seL4 – are written to a model of machine hardware which is remarkably similar to that of the DEC PDP-11 of the 1960s and 1970: a set of homogeneous cores accessing (via MMUs) a common physical address space containing main memory, plus memory-mapped devices, some of which can perform DMA.

This is, of course, a fiction: modern SoCs and server platforms are really a complex network of heterogeneous cores and intelligent devices, many of which are running their own firmware and "operating systems". The *de facto* OS of a modern computer – the thing managing the whole machine – is a motley collection of such cores and their system software, only a small fraction of which runs what is popularly termed "the operating system".

This is an impasse: OSes (including seL4) are designed for nothing more than a homogeneous set of cores with a common view of the address space, and extending them to a complete, real computer is impractical. Hardware designers recognize this near-impossibility, and so add semi-hidden cores running their own firmware to work around the problems. The result is a catastrophe of system design, including a plethora of security exploits like remote over-the-air compromises due to weaknesses in WiFi modem firmware[1]. Crucially, since the *de facto* OS of a modern computer has no overall design (even if it contains well-designed or verified components like microkernels), it is not possible to specify its correct behavior, let alone verify that it conforms to such a specification.

We are building Kirsch, a new OS that solves this problem by embracing and formally capturing the heterogeneity and multiple trust domains of modern hardware. Rather than a clean-slate design, Kirsch acknowledges the need to build a secure OS for a complete machine out of both trusted *and untrusted* parts, accommodating the proprietary firmware on some devices. Instead of trying to extend an existing kernel to a platform totally unsuited to it, Kirsch assembles an OS for a real computer out of components which have *explicit, formally-specified trust relationships* based on the hardware.

In doing so, Kirsch can recover the power of the correctness proofs of seL4 by re-basing them on a clear set of trust assumptions about the underlying hardware. seL4 can thus be turned into a critical component (or set of components) of a well-specified OS for the entire heterogeneous platform, which can deliver strong isolation guarantees to processes and VMs running above it.

The formal hardware model on which Kirsch is based is a decoding net representation [2] of the complete platform, which captures exactly what each physical core or device ("context" in Kirsch parlance) can access, and how its access can be restricted by MMUs or other protection units in the interconnect. This decoding net thus induces a trust relationship between contexts, which is the basis for reasoning about isolation, protection and authorization in the system.

The decoding net representation of a modern platform is highly complex, reflecting the complexity of the hardware itself. To make the OS design, and reasoning about it, tractable, we therefore define a *single, shared, logical address space* for the whole machine, in which every addressable resource is allocated a unique region. Each context has access to a subset of this space. This factoring into privileged components which maintain this address space in each context, and the rest of the OS which runs within it, greatly simplifies the design and reasoning about it.

seL4 fits beautifully into this model: an seL4 instance can run from, and manage, a region of RAM to which all possible access from other devices and cores in the system is explicitly considered and enabled. Above this, the seL4 kernel creates *virtual contexts* (processes or VMs) which benefit from the verified isolation properties of seL4 composed with the explicit underlying trust relations provided by Kirsch. seL4 thus becomes a critical component of a complete OS which securely manages the entire machine, including heterogeneous cores and devices which run proprietary or otherwise untrusted software – assuming that such devices can be verifiably contained. If they can't, the Kirsch toolchain makes it clear why they cannot be isolated, and the developer is given an explicit choice between acquiring better hardware which can deliver the guarantees seL4 and the rest of the OS needs, or making a leap of faith.

[1] See, for example, *Over The Air: Exploiting Broadcom's Wi-Fi Stack*, Google Project Zero, `https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_11.html`

[2] *Putting out the hardware dumpster fire*, Ben Fiedler, Daniel Schwyn, Constantin Gierczak-Galle, David Cock, Timothy Roscoe. HotOS '23: Proceedings of the Workshop on Hot Topics in Operating Systems, June 2023, pp. 46-52.