School of Computer Science & Engineering
Trustworthy Systems Group

Verifying the seL4 Core Platform

Verified Mapping of seL4CP Spec to CapDL
Step 2: Mapping Prototype and Verification Plan

Zoltan A. Kocsis

December 2022

1 Introduction

The **seL4 Core Platform** (seL4CP) is a new operating system for embedded systems, based on the high-assurance seL4 operating system kernel. Unlike previous frameworks, such as CAmkES, which require a deep understanding of seL4 systems development, seL4CP has ease-of-deployment as its core aim, and is intended to be accessible to all embedded-systems developers.

The seL4 Core Platform was co-developed with a device called Laot that protects critical infrastructure (such as remotely-managed power stations) from cyber attacks. The Laot device was build using seL4CP by an engineer without prior seL4 systems development experience, which indicates that the usability aim had been successfully achieved.

CapDL is a language for describing access rights in seL4-based systems. CapDL specifications can be used to track which objects and entities have access to which seL4 capabilities, and provide complete descriptions of the capability distribution in a system running on the seL4 kernel. Moreover, CapDL specifications can be used as input to security analysis tools, including tools that help automate system bootstrapping processes. This makes CapDL an extremely powerful and versatile tool for managing seL4-based systems.

The Trustworthy Systems group, during the course of the DARPA Cyber Assured Systems Engineering (CASE) research project, has developed a *formally verified system initialiser* (`case-init` that can place an seL4 system into any CapDL-specified state.

Since the old CAmkES framework had a verified mapping from its architecture description language to CapDL, CAmkES projects can take advantage of the assured system initialisation provided by `case-init`. However, the System Description Format used by the Core Platform currently lacks an analogous CapDL mapping, so these benefits are not yet available to projects targeting seL4CP.

The goal of this project is to develop a verified mapping from the **SDF** format used by seL4CP to CapDL, so that seL4CP-based systems can take advantage of the assured system initialisation provided by system initialiser developed as part of the CASE project. This will requires developing new tooling to generate the necessary CapDL specifications from Core Platform system descriptions.

2 SDF-CapDL Translator

An seL4CP system is constructed from a collection of ELF binaries that will run in each protection domain, as well as an XML *System Description Format* (SDF) configuration file. This configuration file describes the protection domains present in the system, and the channels between them (i.e. which protection domains are allowed to interact with each other), as well as interrupts and shared memory regions. The seL4CP manual contains a thorough description of the XML configuration format.

2.1 Development

The first proof-of concept implementation, developed in part by Kevin Tran, was the starting point for the development of the current, more fully-featured SDF-to-CapDL translator.

The proof-of-conceptonly supported projects involving the `seL4cp` features in the example project for the ZCU102 board: a single protection domain, no handling of IRQs, protected procedure calls or memory regions. The program could automatically generate a CapDL spec based on a user-provided SDF system specification that only included the supported features. The generated CapDL spec was also not complete enough to boot a seL4CP-based system using CapDL loader.[1]

The presentversion has the following additional features:

- Full support for 64 protection domains and channels between them.
- The capabilities for IRQs are handled properly, in accordance with the Milestone 1 specification.
- The capabilities for endpoints involved in protected procedure calls are handled properly.
- Some support for memory regions is available.

2.2 Approach

The `sel4cp-tool` (also referred to as the seL4 Core Platform SDK) is a command-line Python tool that processes a user-provided SDF system description file and a set of user-provided ELF binaries, and generates a complete system image suitable for loading by the target platform bootloader.

Our current implementation of the SDF-CapDL mapping augments the `sel4cp-tool` for with automatic generation of CapDL language output based on the provided system specification. The decision to augment the SDK itself allows us to reuse its facilities for interacting with SDF system specifications; moreover, it allowed us to rely on the seL4 toolchain for generating, parsing and loading CapDL specifications of systems (`capdl-tool`), which also has preexisting Python CapDL bindings. These bindings allow us to build up an in-memory database storing the same information as a CapDL spec, and emit a `cdl` file as a loadable, syntactically correct CapDL specification. Note that the Python CapDL bindings do not come equipped with CapDL semantics: it's perfectly possible to generate unbootable or invalid systems using them. As it stands, the current Python implementation is intertwined with the Core Platform SDK and the implementation itself is not directly amenable to formal verification.

[1]The CapDL loader is the standard, unverified initializer, distinctint from the formally verified CASE initializer.

2.3 Using the CapDL generator

If the seL4CP build environment is set up correctly, running the `build.sh` script in each example directory will produce an `examplename.cdl` file in the build directory. This can be compared with the `report.txt` generated by the `sel4cp-tool`, which contains the boot instructions generated for the old initializer, the `sel4cp-monitor`. One can compare these manually with the generated CapDL to ensure that the two systems build into similar initial states (modulo differences in fault handling / monitor behavior).

Note that the seL4 Core Platform is built exclusively for the MCS version of the seL4 microkernel, while the CASE loader does not yet support the MCS variant. This means that the generated CapDL files currently cannot be used to initialize an seL4CP system in a verified way. However, we can evaluate the correctness of the generated files by using the regular, unverified CapDL loader to initialize the system. We have tested and have been able to initialize systems using the generated `hello.cdl` and `timer.cdl` files, and run them on the `zcu102` in the Trustworthy Systems machine queue.

Engineering Difficulties

Our efforts to initialize systems using the CapDL loader and the `cdl` files generated by the `sel4cp-tool` uncovered a likely toolchain issue affecting seL4 Core Platform builds, which resulted in systems that were unable to boot. The causes are not entirely clear, and merit further investigation. Currently, we pin all our cross-compilers to specific versions using a `nix develop` environment to avoid triggering this issue.

3 Verification Plan

3.1 Accomplishment: Off-by-one errors in seL4CP

The effort to formally specify and implement the SDF to CapDL translation has led to the detection of an off-by-one error in the current seL4CP implementation and documentation. The manual stated that the system supports a maximum of 64 channels and interrupts per protection domain, but this is not true. Due to the way the capability badging of notifications is implemented, only 63 channels are supported. In the current seL4 Core Platform implementation, the SDK does not properly enforce the limit of 63 channels per protection domain, and it is possible for a user to exceed this limit by one without receiving any errors or warnings. This can lead to unpredictable behavior and potential errors in the system. This issue was detected thanks to the formal specification of the SDF to CapDL translation. The fix will involve either a change to the manual (to correct the phrase), or a slight change to the capability badging so that we can support 64 channels.

3.2 Challenge: CASE Initializer MCS Support

Through the DARPA CASE project, TS has developed a verified system initializer that can boot the system into any state defined by CapDL. By creating a verified mapping from SDF to CapDL, we can ensure that an seL4CP system boots up into the state described by the spec. Unfortunately, the CASE initializer does not support all of the required features for seL4CP systems. In particular, it lacks support for the mixed-criticality scheduling extensions. Meanwhile, the seL4 Core Platform *only* supports the MCS kernel.

Some possible options for addressing the lack of MCS support in the CASE initializer are:

- Extend the CASE initializer to support MCS, so that it can boot seL4CP systems with MCS kernels. This will require additional proof work, but will allow us to use the CASE initializer and the full capabilities of the seL4 Core Platform. This is the longer-term solution, but it will require consulting with the developers of the original CASE initializer to get accurate estimates of the required proof effort.
- Temporarily use a version of seL4CP that does not require MCS, e.g. by using a fork of a very old versions of the platform that does not rely on MCS support. This would allow us to continue using the CASE initializer as is, but would limit the capabilities of our system and is certainly not a long-term solution. Moreover, the support for our development boards would have to be backported to the older version. However, this requires significantly lesser development effort than changing the proofs, and might allow us to progress towards access control policy refinement (see below) without getting blocked by the MCS-compatible CASE initializer development.

In the meantime, we will make sure to retain support the (unverified) CapDL loader: this will allow developers to load their systems using CapDL, and migrate to the CASE initializer as soon as its MCS version becomes available. Note that retaining support for the unverified loader would be required anyway, as the CASE initializer build process is too slow for development and should therefore only be used for producing a production system.

3.3 Access Control Policy Refinement

The CASE initializer proofs go higher than merely showing that the initializer correctly sets up the capability distribution of the system according to some CapDL.

An Access Control Policy can be derived from a CAmkES system specification, and an Access Control Policy can be derived for the initialized system as well. The CASE initializer proofs are able to establish that the Access Control Policy of the system specification is satisfied by the initialized system. We are currently investigating whether we could provide a similar policy preservation proof for seL4CP-based systems.

While the current executable implementation of the SDF-CapDL translator is not formally verified (and not easily formally verifiable), an access control policy refinement proof would work around this issue: we will not have to worry about our the translator being “correct” in some lower level sense, as long as in each particular instance we can prove that the Access Control Policy defined in the SDF is in fact the same as the one defined by the CapDL that will be used to initialize the system. If, however, the current investigation into the access control policy proofs reveals that showing the analogous property for the SDF would be too much effort for the remaining time in this Phase, we can alternatively use the formalized SDF semantics to develop a verifiable CapDL generator directly in ML.