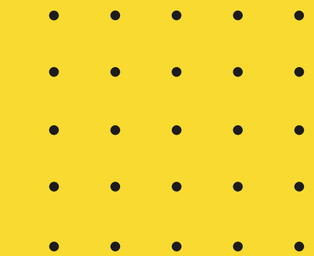


The seL4 Core Platform

Zoltan A. Kocsis (lecturer, UNSW)

seL4 Summit 2022, Munich

Progress

seL4 Summit 2020

- seL4cp revealed
- RFC released

Progress

seL4 Summit 2020

- seL4cp revealed
- RFC released

2020-2022

- seL4cp released
- on Github today
- used in-house and externally

Progress

seL4 Summit 2020

- seL4cp revealed
- RFC released

2020-2022

- seL4cp released
- on Github today
- used in-house and externally

Now

- we're verifying seL4cp

What is the seL4 Core Platform?

"A simple operating system for the seL4 microkernel."

says the Github blurb.

What is the seL4 Core Platform?

"A simple operating system for the seL4 microkernel."

says the Github blurb.

More specifically:

A simple, small operating system for

- embedded systems
- cyber-physical systems
- IoT

built on simple, static architectures.

That's by far the most common form of seL4 deployment in the wild.

What is the seL4 Core Platform?

"A simple operating system for the seL4 microkernel."

says the Github blurb.

More specifically:

A simple, small operating system for

- embedded systems
- cyber-physical systems
- IoT

built on simple, static architectures.

That's by far the most common form of seL4 deployment in the wild.

It helps you do better work via:

- making development and deployment easier and more user-friendly,
- giving you "correct" (secure/safe by default) ways to use seL4 mechanisms,
- keeping a minimal, verified trusted computing base and retaining the seL4 kernel's superior performance

Where does it fit?

MCS

- sel4cp is fully built on the MCS kernel.

Provides

- minimal, reasonable policy (with security/safety/isolation in mind)
- reasonable degree of application portability

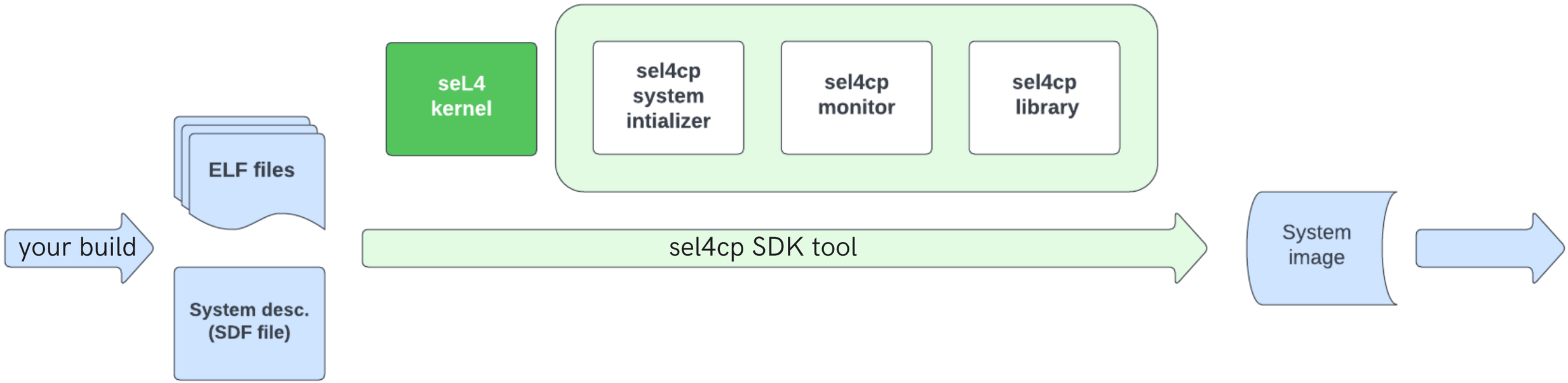
BYOB

Bring Your Own Build system:

- sel4cp sdk integrates with your build system
- build your application ELF in any way you want
- as long as ELF supports the sel4cp binary interface, sel4cp will be able to load and use it

BYOB

an application built on sel4cp



Simple interfaces

an application built on sel4cp

main.c

```c
#include <sel4cp.h>

void init(void)
{
  sel4cp_dbg_puts("initializing \n");
}

void notified(sel4cp_channel ch)
{
  sel4cp_dbg_puts("got notified \n");
  switch(ch) { case 2: /*webcam*/ ... }
}
```

system.sdf

```
<protection_domain name="main" priority="254">
  <program_image path="main.elf" />
</protection_domain>

<memory_region name="cam_buffer" size="0x400000"/>

<protection_domain name="webcam" priority="99">
  <program_image path="webcam.elf" />
  <map mr="cam_buffer" vaddr="0x2000000" perms="rw" />
</protection_domain>

<channel>
  <end pd="main"   id="2" />
  <end pd="webcam" id="1" />
</channel>
```

The parts of sel4cp

sel4cp init task + monitor

Runs as first user task.
Executes invocations to create and configure kernel objects, distribute caps in accordance with system description.

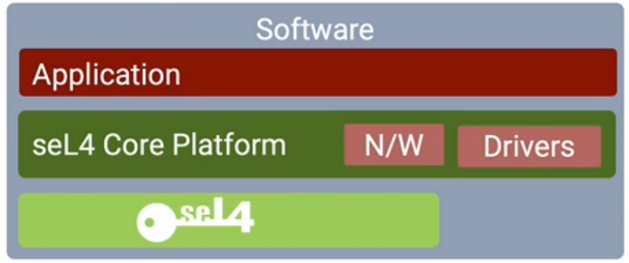
After init acts as the fault handler for protection domains.

(~1.2 kloc)

sel4cp library

Provides functions for handling notifications, memory regions, protected procedure calls (IPC), IRQs.

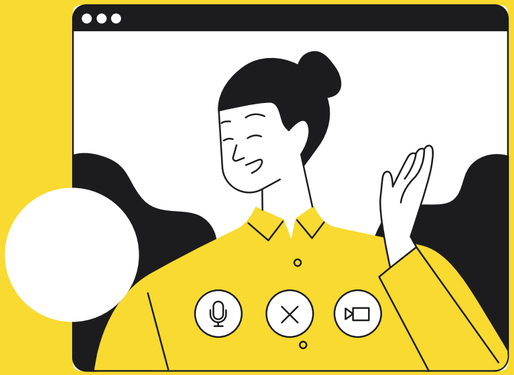
(< 1 kloc)



sel4cp build tool

Integrates with your build system, creates system image based on provided ELF and system description.

So... is it user-friendly?

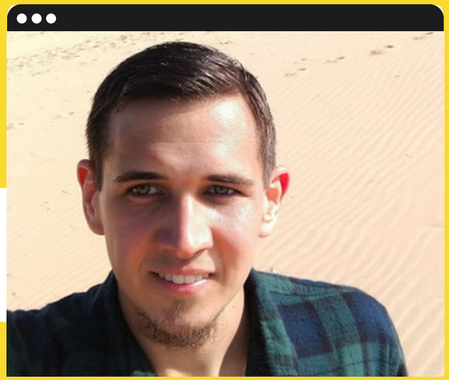


The strictest usability testing on the planet

13 Oct:
you'll experience it yourself at the workshop.

So... is it user-friendly?

13 Oct:
you'll experience it
yourself
at the workshop.

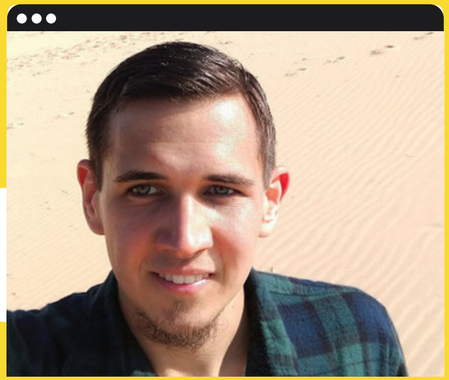


The strictest usability testing on the planet

Even I am able to get it to work!

So... is it user-friendly?

13 Oct:
you'll experience it
yourself
at the workshop.



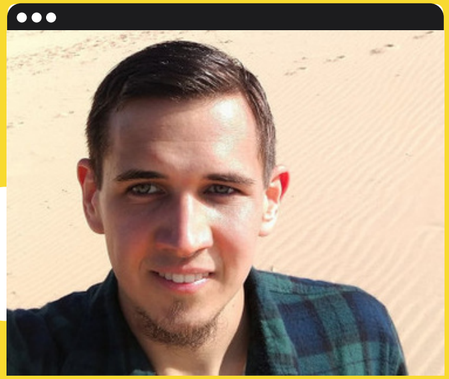
The strictest usability testing on the planet

Even I am able to get it to work!

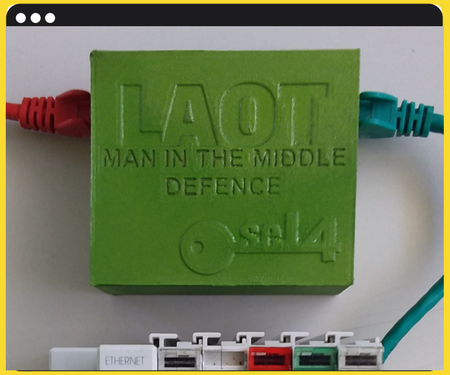
(with a little help from our students)

So… is it user-friendly?

13 Oct:
you'll experience it
yourself
at the workshop.



The strictest usability testing on the planet



Not just usable…

USED

Even I am able to get it to work!

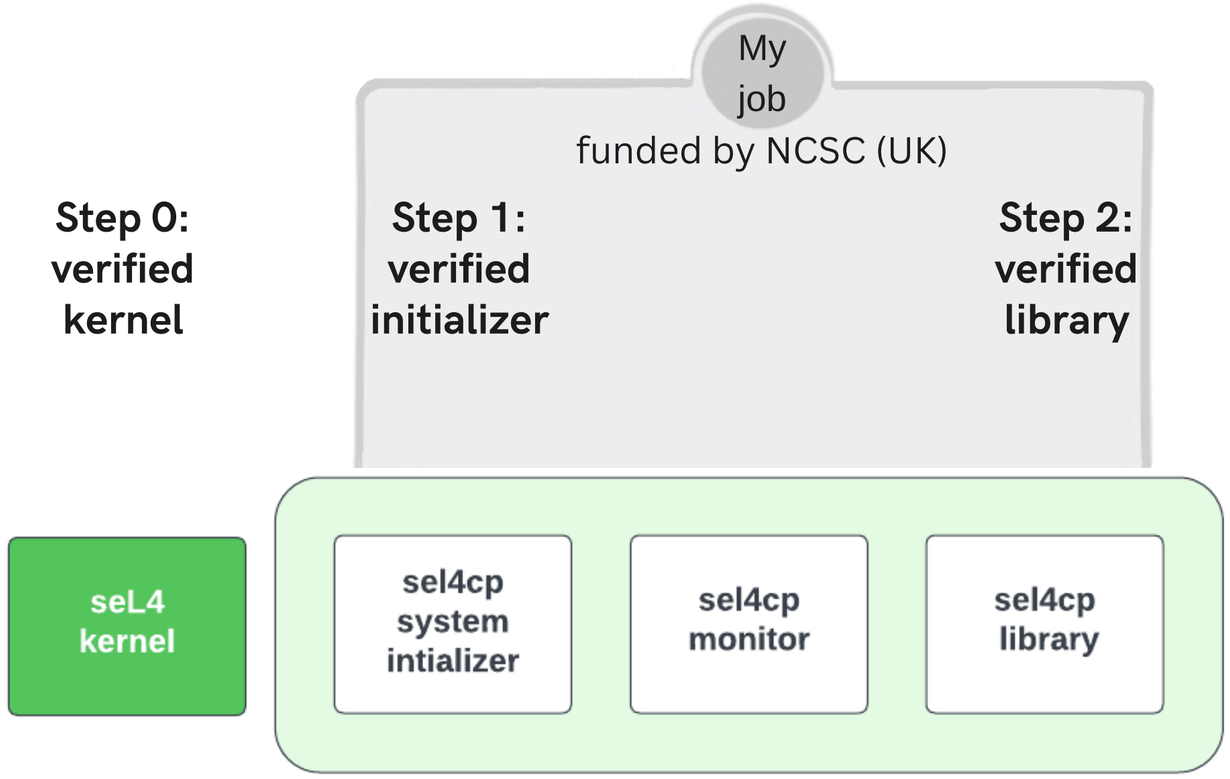
(with a little help from our students)

Developed together with **Laot**: a protective device for critical infrastructure. Funded by ICERA grant from the Australian Department of Defence.
Involved:

- Ben Leslie (Breakaway, sel4cp lead),
- Phil Maker at EDS
- Gernot

https://trustworthy.systems/projects/TS/laot

Two steps to verified sel4cp



Verified init

Leverage pre-existing work

TS, in DARPA's Cyber Assured Systems Engineering (CASE) program, produced a verified **CapDL loader**, which can boot an seL4 system into CapDL-specified states.

Problem

While CAmkES uses CapDL, seL4cp uses its own system description format SDF.

Solution

A verified translation of SDF into CapDL.

Current:

ELF + SDF

From your build system.

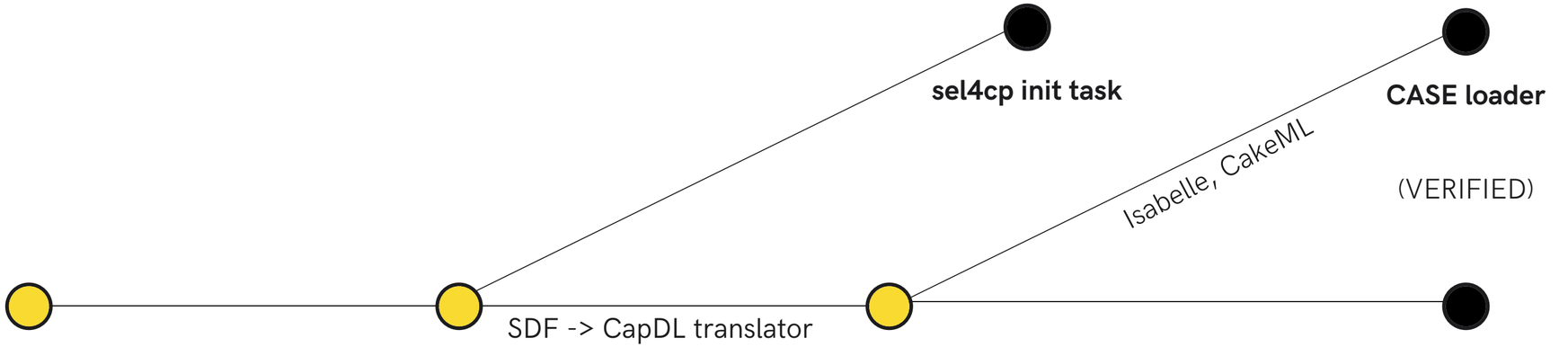
sel4cp SDK

configures the system initialiser task.

sel4cp Init task

starts your system, acts as monitor

Future:



ELF + SDF

From your build system.

sel4cp SDK

Generated CapDL

is checked against the semantics (access policy) of the original SDF.

CapDL loader

(unverified, for test/debug builds)

Roadmap

SDF -> CapDL translator

There is a working version which translates SDF to CapDL, but it doesn't yet support all SDF features.

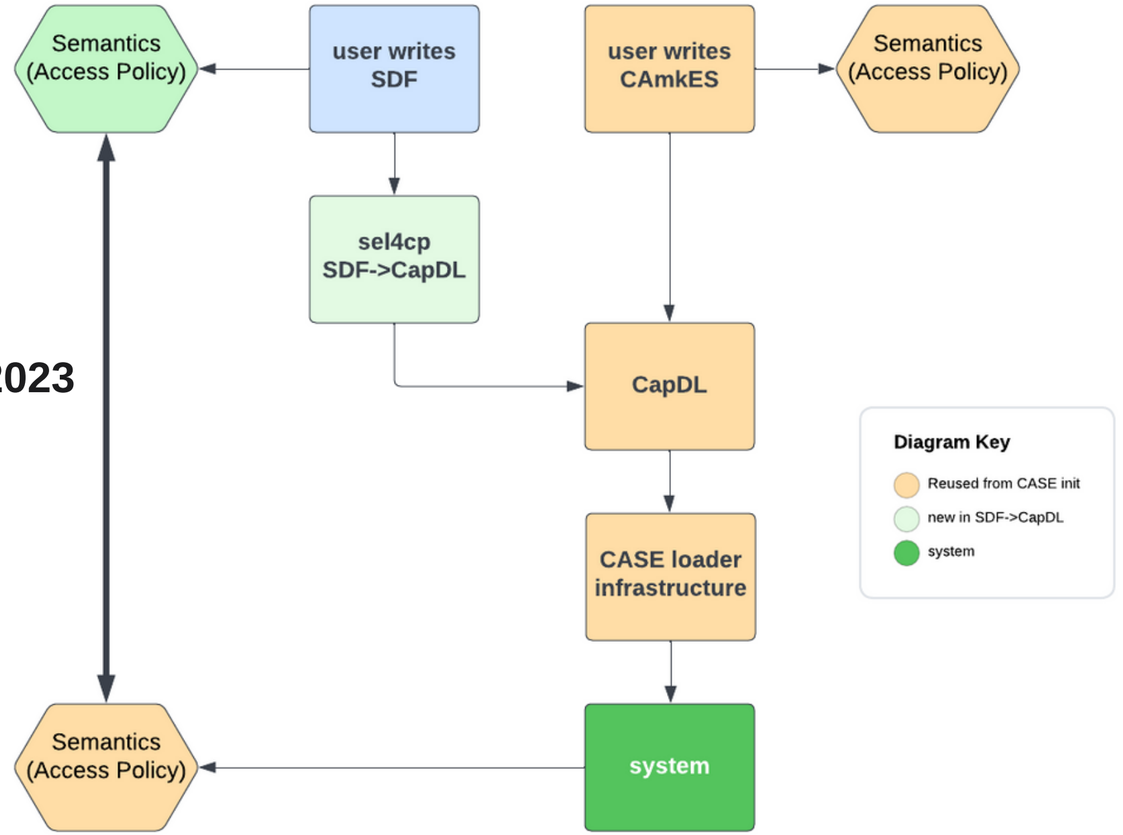
SDF Semantics

informally defined, but not yet formalized

March 2023

CASE loader

exists, but no MCS support yet!
(recall: sel4cp is MCS-only)



Verified library

The library is small

< 1k lines of code,
8 functions in the API,
handler loops,
thin wrappers over kernel calls

Aim: automated verification

Using SMT-based techniques.

No 170k lines of code, no 11 person-years again :)

Some goals:

- Absence of undefined behavior
- Priority-order processing
- Access policy is maintained (will need init semantics)

Status: not yet where I want it to be.
Goal: March 2023

Thank you!

Questions?