- **Regan Robertson** Mod • 15 days ago
  Good Afternoon,
  The video will start on this page at 14:30, and you may have to hit play or unmute. As a reminder this site works best in a Chrome Browser. You can write in comments through this feature to continue the conversation or ask questions.
  - Reply
  - Share ›

Renato Levy • 15 days ago
the use of CIL would be better, because it is a stack machine, which is easier to proof
- 1
- Reply
- Share ›

Kevin Hamlen Renato Levy • 15 days ago
Hi Renato: Yes, proving things at the CIL level is certainly easier; but the guarantees you get are much weaker because you must blindly trust a rather large infrastructure that either interprets or JIT-compiles the CIL code. So the verification gets easier only because you're getting significantly lower assurance.
- Reply
- Share ›

Gernot • 15 days ago

Isn't this exactly the approach we used for doing the binary verification of seL4, proving that the binary is a refinement of the formal, C-level model of seL4?

Reply

Share ›

Kevin Hamlen Gernot • 15 days ago

Hi, Gernot: No, it's not the same because proving that a binary is a refinement of a C-level model assumes that there exists a model that is expressible in C. There are portions of seL4 inexpressible in C (e.g., in-lined assembly), where the approach you're describing can't be applied.

Reply

Share ›

Gernot Kevin Hamlen • 15 days ago

We have a model of the kernel in Isabelle, which happens to be derived from C but that's not relevant to the verification story, the only relevant point is that it's proved

to be a refinement of the abstract spec. Why does it matter for our binary verification approach that it's derived from C code? It might contain formalised assembler code, why would that affect the binary verification proof chain?

Reply

Share ›

Gernot Gernot • 15 days ago • edited

Note, I'm an OS guy, not a formal verification person, so I may misunderstand things...

Reply

Share ›

Kevin Hamlen Gernot • 15 days ago

If your abstract spec and model is at the level of arbitrary assembly code, then yes, I think we're talking about the same approach. But from what I've seen of the part of the seL4 proof you're talking about, the model is not that low-level. It uses a modeling language that cannot express the semantics of many low-level ISA operations, such as those that manipulate the CPU state directly, because it is much harder to prove things in such a low-level model.

Reply

Share ›

Kevin Hamlen Kevin Hamlen • 15 days ago

Here's another way to think about it: The first step of the refine-to-C approach you're talking about is to heuristically decompile the binary to a higher level form that, when compiled, gives you the original binary. The higher level form might be LLVM IR or something similar. But there are portions of these binaries such that there exists no LLVM IR that compiles to them. If you change your model to be expressive enough to recover the target binary exactly, you end up with a model that is not higher level at all; it is the hardware ISA model. Such a low level model is very difficult to use when verifying the non-assembly portions of seL4.

Reply

Share ›

Lennart Beringer • 15 days ago

Myreen's decompiler work (related to seL4..) comes into mind, too.

Reply

Share ›

Jason H Li Lennart Beringer • 15 days ago

I think D61's binary proof leveraged Myreen's decompiler work.

- Reply
- Share ›

Gernot Lennart Beringer • 15 days ago

this is what we used, long-standing collaboration with Myreen

- Reply
- Share ›

Lennart Beringer Gernot • 15 days ago

Yes, Gernot's and my question raced...

- Reply
- Share ›

Renato Levy • 15 days ago

which intermediate code semantics are you using?

- 1
- Reply
- Share ›

Jason H Li Renato Levy • 15 days ago

Thought I heard IL in BAP.

- Reply
- Share ›

Renato Levy Jason H Li • 15 days ago • edited

I meant when bypassing BAP. is it the same?

- Reply
- Share ›

Jason H Li • 15 days ago

Kevin - as FM becomes more feasible but still not really ready for prime time (for developers) ... what is the status of the bottom-up tools for adoption? I asked Lennart about Princeton/Yale Summer schools. Is there anything developers can learn / exercise yours through some means like on-line courses or summer schools?

- 1
- •
- Reply
- •
- Share ›

Lennart Beringer • 15 days ago

What memory model and concurrency primitives does your system support?

- 2
- •
- Reply
- •
- Share ›

Stuart Card • 15 days ago

Potential applicability to "Genetic Improvement of Software" by "Genetic Programming", where the transformations may be arbitrary recombinations and mutations?

- •
- Reply
- •
- Share ›

Arslan Khan • 15 days ago

Q: How do you tackle the challenges with reverse engineering the source code from binary, since current approaches for this seem to be less than perfect?

Q2: How do you compare the effort for proving something semantically same in top-down to bottom-up approach? I am curious because in top-down we have all the type information but in bottom-up that information seems to be missing, right?

2

Reply

Share ›

Regan Robertson Mod • 15 days ago

Please join us for the next session that starts now. You can either go back to agenda to get to the next session or at the bottom of the page there is a next session button.

Reply

Share ›