Master level Internship: Formalizing Hardware Security Mechanisms: interrupts, MMU and SMT proofs

Pierre Wilke
pierre.wilke@centralesupelec.fr
CIDRE-SUSHI team

Context The security of applications eventually depends on the security of all the abstractions layers they are built upon. The application itself may contain some hardening techniques, it may benefit from isolation provided by the operating system (OS), and it may also be secured through the use of hardware security mechanisms. For critical applications, formal methods (e.g. static analysis, model checking, formal proof) have been successfully applied at the source code level (e.g. Astrée, VST), at the compiler level (e.g. CompCert [6]), at the OS level (seL4 [5], CertiKOS [4]), and more recently at the hardware level (e.g. Kami [3], Kôika [2]).

Several of these approaches are based on the Coq proof assistant,[1] which allows for formal definition of semantics and proofs. The Coq proof assistant is a program that helps building proofs and rigorously checks that the proofs are correct. The Kôika language is a high-level hardware design language (HDL) embedded in Coq, together with a verified compiler to Verilog. Thanks to this approach, one can design a circuit in the high-level HDL (Kôika) and generate from that a description in a low-level HDL (Verilog) that standard hardware tools can use to synthetise the circuit on FPGA. The authors proved that the compiler preserves the semantics of Kôika.

In the SUSHI group, we aim at proving security properties guaranteed by hardware/software mechanisms. We proposed an approach to prove security properties about Kôika designs. This requires to compile Kôika designs to an intermediate representation more suitable than the Kôika representation to formal verification. We proved that this compilation is correct and successfully apply this methodology to prove the security of a shadow stack[2] in a RISC-V processor [1]. This forms a foundation for proving more complex security mechanisms.

Our current approach consists in automatically compiling Kôika designs to a more explicit representation, and then manually proving the properties of interest on this representation. The manual proof effort for this second step is still very high, and specific to each property and security mechanism. In order to automate this last step, we have started to leverage SMT (Satisfiability Modulo Theory) solvers. This approach looks very promising : we have managed to prove automatically the security properties about our shadow stack described in [1].

Internship The goal of the internship is to design and prove other security mechanisms in the RISC-V processor written in the Kôika language. This will also be the opportunity to contribute to better integrate the SMT solver within the proof workflow in Coq.

Here are the main steps of the internship:

- get familiar with Coq and the Kôika language;
- get familiar with the RISC-V processor written in that language[3] and the simulation and synthesis toolchains;
- extend the processor with new mechanisms (e.g. forward-edge Control-Flow Integrity or memory isolation)
- state the security properties that these mechanisms should enforce;
- prove these properties;
- integrate with the SMTCoq [4] plugin.

Required skills or interests The candidate should have familiarity with at least one of the following:

- hardware design languages (e.g. Verilog/VHDL) and computer architecture;
- or formal methods (e.g. Coq, SMT solvers).

[1] https://coq.inria.fr
[2] https://en.wikipedia.org/wiki/Shadow_stack
[3] https://gitlab.inria.fr/cidre-public/koika-llr/-/blob/main/examples/rv/RVCore.v
[4] https://smtcoq.github.io/

Institute The internship will take place at CentraleSupélec in Rennes, France, in the soon-to-be-created SUSHI Inria team[5]. This team is part of the IRISA laboratory.[6] The internship will be advised by Pierre Wilke and Guillaume Hiet.

Practical aspects The internship will last 5 months, starting from February, 2024. The intern will receive a "gratification" of 609€ per month. Housing options may be available on campus, or close to the campus.

This internship could be followed by a Ph.D. thesis on similar subjects.

References

[1] Matthieu Baty et al. "A Generic Framework to Develop and Verify Security Mechanisms at the Microarchitectural Level: Application to Control-Flow Integrity". In: *36th IEEE Computer Security Foundations Symposium, CSF 2023, Dubrovnik, Croatia, July 10-14, 2023*. IEEE, 2023, pp. 372–387. DOI: 10.1109/CSF57540.2023.00029. URL: https://doi.org/10.1109/CSF57540.2023.00029.

[2] Thomas Bourgeat et al. "The essence of Bluespec: a core language for rule-based hardware design". In: *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*. Ed. by Alastair F. Donaldson and Emina Torlak. ACM, 2020, pp. 243–257. DOI: 10.1145/3385412.3385965. URL: https://doi.org/10.1145/3385412.3385965.

[3] Joonwon Choi et al. "Kami: a platform for high-level parametric hardware specification and its modular verification". In: *Proc. ACM Program. Lang.* 1.ICFP (2017), 24:1–24:30. DOI: 10.1145/3110268. URL: https://doi.org/10.1145/3110268.

[4] Ronghui Gu et al. "CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels". In: *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*. Ed. by Kimberly Keeton and Timothy Roscoe. USENIX Association, 2016, pp. 653–669. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/gu.

[5] Gerwin Klein et al. "seL4: formal verification of an operating-system kernel". In: *Commun. ACM* 53.6 (2010), pp. 107–115. DOI: 10.1145/1743546.1743574. URL: https://doi.org/10.1145/1743546.1743574.

[6] Xavier Leroy and Sandrine Blazy. "Formal verification of a C-like memory model and its uses for verifying program transformations". In: *Journal of Automated Reasoning* 41.1 (2008), pp. 1–31. URL: http://xavierleroy.org/publi/memory-model-journal.pdf.

[5] The team is currently CIDRE: https://team.inria.fr/cidre/

[6] https://www.irisa.fr/en