Master level Internship: Formalizing RISC-V Enclaves

Guillaume Hiet, Pierre Wilke, Jan Tobias Mühlberg
`guillaume.hiet@centralesupelec.fr`
`pierre.wilke@centralesupelec.fr`
`jan.tobias.muehlberg@ulb.be`

Context The security of applications eventually depends on the security of all the different layers of abstractions involved in the application. 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 [7]), at the OS level (seL4 [6], CertiKOS [5]), and more recently at the hardware level (e.g. Kami [4], Kôika [3]).

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. Based on this approach, one can design a circuit in the high-level HDL (Kôika) and compile that to a hardware description in a low-level HDL (Verilog) that standard hardware tools can use to synthetise the circuit or to program an FPGA, while preserving the semantics of Kôika.

The SUSHI group at Inria/CentraleSupélec aims at proving security properties guaranteed by hardware/software mechanisms. They 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. In recent publications from the SUSHI group, the authors prove that this compilation is correct. They successfully apply this methodology to prove the security of a shadow stack[2] in a RISC-V processor [2]. 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, which looks very promising: we have managed to automatically prove the security properties about our shadow stack described in [2].

In a next step we seek to implement advanced software isolation and additional security primitives that are commonly provided by Trusted Execution Environments (TEEs, [9]) in Kôika, further extend this with support for availability, secure real-time processing and mixed-criticality support [1], and prove properties regarding software isolation and secure exception handling for the resulting hardware design. Hardware support for strong notions of availability in TEEs have earlier been developed by a team of researchers including Jan Tobias Mühlberg (now at Université Libre de Bruxelles) based on the Sancus processor [8], without mechanised proofs.

Internship The goal of the internship is to implement TEE-like software isolation with support for real-time execution on the RISC-V processor developed in the Kôika language. The final objective is to leverage the Coq proof assistant to mechanize the security and availability guarantees offered by the resulting processor design.

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 and the proof methodology that we used in [2];
- get familiar with software isolation and availability support in Sancus and Aion [8, 1];
- modify the RISC-V processor in Kôika to implement an enclave mechanism inspired by Sancus;
- state the desired security properties;
- prove them by relying on the recent development around SMT solvers

[1] `https://coq.inria.fr`
[2] `https://en.wikipedia.org/wiki/Shadow_stack`

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).

Institute The internship will take place at CentraleSupélec in Rennes, France, in the soon-to-be-created SUSHI Inria team[3]. This team is part of the IRISA laboratory.[4]

The internship will be advised by Pierre Wilke and Guillaume Hiet from CentraleSupélec, and by Jan Tobias Mühlberg from Université Libre de Bruxelles.

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] Fritz Alder et al. "Aion: Enabling open systems through strong availability guarantees for enclaves". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 1357–1372.

[2] 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`.

[3] 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`.

[4] 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`.

[5] 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`.

[6] 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`.

[7] 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`.

[8] Job Noorman et al. "Sancus 2.0: A low-cost security architecture for iot devices". In: *ACM Transactions on Privacy and Security (TOPS)* 20.3 (2017), pp. 1–33.

[9] Moritz Schneider et al. "Sok: Hardware-supported trusted execution environments". In: *arXiv* (2022).

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

[4] `https://www.irisa.fr/en`