Building and Verifying a Simple Secure Compiler

General Information

Advisor: Cătălin Hriţcu ⟨catalin.hritcu@inria.fr⟩

Institition: Inria Paris, Prosecco Team

Inria Team leader:
Karthikeyan Bhargavan ⟨karthikeyan.bhargavan@inria.fr⟩

Location: 2 rue Simone Iff, 75012 Paris, France

Language: English

Topics: security, verification, programming languages, compilation, full abstraction

Existing skills or strong desire to learn:

- security foundations and/or building secure systems;
- formal verification (e.g. in Coq or F⋆).

Context

Severe low-level vulnerabilities abound in today's computer systems, allowing cyber-attackers to remotely gain full control. This happens in big part because our programming languages, compilers, and architectures were designed in an era of scarce hardware resources and too often trade off security for efficiency. The semantics of mainstream low-level languages like C is inherently insecure, and even for safer languages, establishing security with respect to a high-level semantics does not guarantee the absence of low-level attacks. *Secure compilation* using the coarse-grained protection mechanisms provided by mainstream hardware architectures would be too inefficient for most practical scenarios.

This new ERC project is aimed at leveraging emerging hardware capabilities for fine-grained protection to build the first, efficient secure compilers for realistic programming languages, both low-level (the C language) and high-level (ML and a dependently-typed variant). These compilers will provide a secure semantics for all programs and will ensure that high-level abstractions cannot be violated even when interacting with untrusted low-level code. To achieve this level of security without sacrificing efficiency, our secure compilers will target a *tagged architecture* [1, 2], which associates a metadata tag to each word and efficiently propagates and checks tags according to software-defined rules. We will experimentally evaluate and carefully optimize the efficiency of our secure compilers on realistic workloads and standard benchmark suites. We will use property-based testing and formal verification to provide high confidence that our compilers are indeed secure. Formally, we will construct machine-checked proofs of *full abstraction* with respect to a secure high-level semantics [3, 4]. This strong property complements compiler correctness and ensures that no machine-code attacker can do more harm to securely compiled components than a component in the secure source language already could.

Objective of the internship

We are looking for highly motivated MSc students interested in getting an initial contact with research. Successful internships with us normally result in a research publication at a good international conference, and most often the MSc students we advise continue with a PhD.

The objective of this internship is to build a simple secure compiler and to prove its security formally. This compiler will translate programs from a core imperative language with procedures and components to an idealized RISC machine, so that each component is protected from the others. Formally, we will investigate two properties capturing secure compilation: The first one is a variant of *full abstraction* that ensures the preservation of confidentiality and integrity properties from the source to the target. The second one, is a new property we call *robust compilation* that captures only the preservation of safety (and thus certain forms of integrity, for instance data invariants) in an adversarial low-level context. While weaker than full abstraction, we expect robust compilation to be easier to enforce efficiently and also to be unaffected by side-channel attacks. Still, expressing robust compilation formally is an open research problem we will have to solve. We will explore different ways for proving secure compilation including interaction traces and logical relations and we will build machine-checked proofs in Coq or F⋆. For details please get in contact using information above.

References

[1] A. Azevedo de Amorim, M. Dénès, N. Giannarakis, C. Hriţcu, B. C. Pierce, A. Spector-Zabusky, and A. Tolmach. Micro-Policies: Formally verified, tag-based security monitors. *Oakland S&P*. 2015.

[2] U. Dhawan, C. Hriţcu, R. Rubin, N. Vasilakis, S. Chiricescu, J. M. Smith, T. F. Knight, Jr., B. C. Pierce, and A. DeHon. Architectural support for software-defined metadata processing. *ASPLOS*. 2015.

[3] Y. Juglaret, C. Hriţcu, A. Azevedo de Amorim, B. Eng, and B. C. Pierce. Beyond good and evil: Formalizing the security guarantees of compartmentalizing compilation. *CSF*. 2016.

[4] Y. Juglaret, C. Hriţcu, A. Azevedo de Amorim, B. C. Pierce, A. Spector-Zabusky, and A. Tolmach. Towards a fully abstract compiler using Micro-Policies: Secure compilation for mutually distrustful components. Technical Report, arXiv:1510.00697, 2015.