18-months Research-Engineer Position

Program Transformation for Information Flow Analysis of C Programs

Keywords: software security, information flow, code generation, *Frama-C*, *C*, *OCaml*, static analysis, monitoring.

Context: CEA LIST, Software Security Labs

The Software Security Laboratory (LSL) has an ambitious goal: help designers, developers and validation experts ship high-confidence systems and software. Objects in our surroundings are getting more and more complex, and we have built a reputation for efficiently using formal reasoning to demonstrate their trustworthiness. Within the CEA LIST Institute, LSL is dedicated to inventing the best possible means to conduct formal verification. We design methods and tools that leverage innovative approaches to ensure that real-world systems can comply with the highest safety and security standards. And in doing so, we get to interact with the most creative people in academia and the industry.

Our organizational structure is simple: those who pioneer new concepts are the ones who get to implement them. We are a fast-growing thirty-person team, and your work will have a direct and visible impact on the state of formal verification.

CEA LIST's new offices are located at the heart of Campus Paris Saclay, in the largest European cluster of public and private research.

Work Description

LSL is the main developer of *Frama-C* [5] (`http://frama-c.com`), a code analysis platform for C programs which provides several collaborative analyzers as plug-ins. *Frama-C* itself is developed in *OCaml*. *Frama-C* allows the user to annotate *C* programs with formal specifications written in the *ACSL* specification language [2]. *Frama-C* can then ensure that a *C* program satisfies its formal specification by relying on weakest preconditions calculus or abstract interpretation.

In the software security research area, information flow analysis [4] is a well-known technique to ensure that sensitive data cannot leak onto an unauthorized channel and thus cannot be used by an attacker. However, applying this approach to large *C* programs is still challenging because of the combination of the low-level intricate semantics of *C* programs and the difficult verification of high-level security properties.

In previous works [1], we handled the issue of information flow for programs with pointers and aliasing and designed a program transformation which encodes information flow of the initial program into the generated one. In that way, it becomes possible to verify a non-interference property on the initial program by verifying several simpler safety properties on the generated one. Furthermore, the best way to perform the verification may be chosen by the end-user (WP, abstract interpretation, monitoring, or a combination of those ...).

Despite this theoretical advance and while a first research prototype has been implemented, a large amount of work remains to transforming this research prototype to an usable tool. From a theoretical point of view, our program transformation relies on a typed model which does not handle some *C* constructs like

heterogeneous casts (*e.g.* from integer to pointer) and polymorphic expression *à la C* (`void *`). From a practical point of view, the current prototype has currently several limitations (variadic functions, function pointers, ...) which have few theoretical difficulties but are not so easy to handle in practice, while the generated code could be optimised during the program transformation.

The goal of the research engineer will be to transform the research prototype to a tool which can be used on real-world *C* programs. To reach this goal, (s)he would have to address the existing limitations by finding both theoretical and pragmatic solutions, and to design and implement new static analysis to optimise the generated code. The work will be challenged against a realistic case study.

References

[1] Mounir Assaf, Julien Signoles, Éric Totel, and Frédéric Tronel. Program transformation for non-interference verification on programs with pointers. In *the 28th IFIP TC-11 International Information Security and Privacy Conference (SEC 2013)*, pages 231–244. Springer, 2013. Best student paper award.

[2] Patrick Baudin, Jean-Christophe Filliâtre, Claude Marché, Benjamin Monate, Yannick Moy, and Virgile Prevosto. *ACSL: ANSI/ISO C Specification Language. Version 1.8*, March 2014.

[3] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C, A software Analysis Perspective. In *Software Engineering and Formal Methods (SEFM)*, October 2012.

[4] Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Commun. ACM*, 20(7):504–513, July 1977.

[5] Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-c: A software analysis perspective. *Formal Aspects of Computing*, pages 1–37, 2015. Extended version of [3].

Applications

Knowledge in several of the following fields is required:

- information flow security
- semantics of programming languages (in particular, the ISO *C* 99 programming language)
- compilation techniques
- formal specification
- functionnal programming (in particular, *OCaml* programming)

Contact: Julien Signoles (*julien.signoles*@cea.fr)