Activity Report 2023

Team EPICURE

Semantic Analysis and Compilation for Secure Execution Environments

Joint team with Centre Inria de l'Université de Rennes

D4 – Language and Software Engineering

Team EPICURE IRISA Activity Report 2023

Contents

Project-Team EPICURE

Creation of the Project-Team: 2022 June 01

Keywords

Computer sciences and digital sciences

A2.1. – Programming Languages

A2.2. – Compilation

A2.2.1. – Static analysis

A2.2.5. – Run-time systems

A2.2.9. – Security by compilation

A2.4. – Formal method for verification, reliability, certification

A2.4.1. – Analysis

A2.4.3. – Proofs

A4.4. – Security of equipment and software

A4.5. – Formal methods for security

Other research topics and application domains

B6.1.1. – Software engineering

B6.4. – Internet of things

B6.6. – Embedded systems

1 Team members, visitors, external collaborators

Research Scientists

- Thomas Jensen [Team leader, INRIA, Senior Researcher, HDR]
- Frédéric Besson [INRIA, Researcher]
- Simon Castellan [INRIA, Researcher]
- Benoit Montagu [INRIA, Researcher]
- Alan Schmitt [INRIA, Senior Researcher, from Apr 2023, HDR]

Faculty Members

- Sandrine Blazy [UNIV RENNES, Professor, HDR]
- Delphine Demange [UNIV RENNES, Associate Professor]
- Benjamin Farinier [UNIV RENNES, Associate Professor]
- Thomas Genet [UNIV RENNES, Professor, HDR]

PhD Students

- Santiago Bautista [ENS RENNES, until Aug 2023]
- Santiago Bautista [ENS Rennes, ATER, from Sep 2023]
- Clément Chavanon [INRIA, from Sep 2023]
- Alexandre Drewery [INRIA, from Sep 2023]
- Jean-Loup Hatchikian-Houdot [INRIA]
- Romeo La Spina [ENS RENNES]
- Tony Law [UNIV RENNES]
- Théo Losekoot [UNIV RENNES]
- Gautier Raimondi [Inria]
- Vincent Rebiscoul [UNIV RENNES, until Nov 2023]
- Malo Revel [UNIV RENNES, from Sep 2023]

Technical Staff

- Aurore Alcolei [INRIA, Engineer, from Oct 2023]
- Pierre Lermusiaux [INRIA, Engineer]
- Louis Noizet [INRIA, Engineer, from Oct 2023]

Interns and Apprentices

- Clément Chavanon [INRIA, from Feb 2023 until Jul 2023]
- Alexandre Drewery [ENS RENNES, until Jun 2023]
- Léo Juguet [UNIV RENNES, from Jun 2023 until Aug 2023]

Administrative Assistant

- Lydie Mabil [INRIA, from Oct 2023]

2 Overall objectives

The security of the software that surrounds us is, more than ever, a scientific challenge of utmost societal importance. More and more software is produced to operate on an increasingly varied number of devices and to provide increasingly complex functionality. There is a pressing need to provide the science and technology for engineering this software so that it becomes safe and secure, in addition to providing the desired functionality. This need is not new and a multitude of programming languages, semantic theories, formal methods, verification tools and techniques have been developed and contribute to meet this need. One of the challenges with this state of affairs is exactly the multitude of languages in which to express the algorithms that we develop, and in particular the distance between those languages for which it is comparatively easy to develop correct software, and those that actually get executed in our computers, telephones, pace makers, cars, smart home IoT devices *etc.*.

No one single silver bullet will solve the problem of developing secure software worthy of the user's trust. We are however convinced that a cornerstone of the answer is *programming language semantics*, *i.e.*, a mathematically robust yet flexible formalism for defining the behaviour of a program. The goal of the EPICURE project is to contribute with semantics-based methods for producing safe and secure software by

- defining new semantic frameworks that will provide more accurate models of modern execution platforms, and which can facilitate the semantic definition of the above-mentioned multitude of programming languages,
- designing formally verified analysis and compilation schemes, with the specific aim of being able to analyse and verify properties of programs written in high-level languages, and to compile both program and the verified properties down to low-level executable representations,
- demonstrate the impact of language-based tools on software security by showing how they can improve the correctness, safety and security of critical software found in modern execution environments, such as the Java virtual machine, the Tezos blockchain written in OCaml, and small operating systems for the IoT such as RIOT.

3 Research program

The overall goal of the EPICURE project is to guarantee the security and safety of key software components of execution platforms, including those used in the IoT and blockchains. Our contribution to this goal will be to develop semantics-based, formally verifiable program analyses and compilation techniques for improving and enforcing software security and safety. The main open challenges in the field include:

- providing mechanised formalisations of modern programming languages (such as Rust, JavaScript, Web Assembly) which facilitate the reasoning about these languages and their tools,
- faithfully modeling architectures on which they execute, taking into account features such as out-of-order execution and trust-enhancing mechanisms such as enclaves and trust zones,
- designing program processing tools such as analyses and compilers, the correctness of which can be verified mechanically,
- developing scalable analyses for proving security properties of high-level programs, and compiling programs and their proofs down to low-level executables, the security of which is guaranteed by the compilation process.

The EPICURE project is structured into the following research axes:

- Semantics and their mechanisation.
- Program analysis.
- Trustworthy compilation.
- Secure execution platforms.

The axis on semantics and their mechanisation will investigate frameworks for defining semantics, in particular the recently proposed *skeletal* semantics and the notion of causal semantics. We will pay particular attention to the semantics of intermediate representations used in compilers and to the semantic description of low-level languages, *e.g.* eBPF. In the axis on program analysis, we plan to conduct work both on the foundations of static analysis and abstract interpretation and on the development of specific analyses, in particular for higher-order polymorphic functional programs. A special attention will be given to the problem of translating results of an analysis from a high-level language to its compiled (low-level) version. In the strand on trustworthy compilation we will pursue the effort on mechanised verification of optimising compilers. We will also examine the security impact of compilation with respect to different (passive and active) attacker models. The intended application areas for these techniques are the Internet of Things and high-assurance block chains.

4 Application domains

The intended application of the scientific results outlined in the previous sections is to improve the safety and security of execution platforms, taken in a broad sense ranging from virtual machines to hardware processors. We will improve on analyses and compilation techniques for verifying and producing safer code, as we will improve on the key software tools and components that implement the execution platform. In this section we outline a number of more concrete applications that we intend to investigate.

4.1 Internet of Things

The Internet of Things offers a large and diverse domain of application for our formal methods. The limitations of the devices populating the IoT mean that a different kind of algorithms are deployed but the security and privacy concerns remain, and are even accentuated by the relative weak protection mechanisms offered by the underlying hardware. In particular, the IoT relies on cryptographic primitives for secure communication and software updates but these primitives are often different from what is used on standard execution platforms due to the limited computing resources. The question of secure compilation and the techniques that we expect to develop can be transferred to the IoT but the security properties might be harder to verify because of optimisations.

On the application level, the distributed and asynchronous nature of the IoT has led to new programming paradigms and novel uses of existing languages (such as JavaScript) that pose new verification challenges, in particular the verification of coordinating programs written in different complex languages in a multitier framework. A multitier language unifies within a single formalism and a single execution environment the programming of the different tiers of distributed applications. On the web, this paradigm unifies the client tier, the server tier, and the database tier. We thus want to investigate how our techniques can be brought to bear on multitier programming languages. In particular, we propose to investigate the design of program analyses for a multitier language for the IoT.

4.2 High-assurance blockchains

Because they enable the distributed management of virtual assets—such as property rights, proofs of payments— blockchain systems play a growing, *critical* role in our societies. Blockchain-based systems, like Ethereum or Tezos, are equipped with so-called *contracts*. A contract is a program which is executed by a *virtual machine* (VM) over the blockchain. The effect of a contract is to update values and assets stored in the blockchain. Thus, any failure in the safety, availability, or security in the VM of a system like Tezos could have dramatic consequences on industries, on public infrastructures, and eventually on people. The pieces of code that lie at the foundations of the Tezos system are entrusted with the safety and security of all the managed assets. The Tezos core software is thus expected to attain the highest levels of clarity and quality, and to get as close as possible to zero defects. This is where formal methods—and in particular static analyses—can help, by giving guarantees about the dynamic behaviour of programs, in an automatic way. The expressive type system of OCaml—the implementation language of Tezos—already provides static safety guarantees by ensuring data is used in a consistent way. In collaboration with Nomadic Labs, we will provide OCaml programs with additional guarantees, by answering questions such as *"can a program raise an exception?"*, *"can a program break some user-defined invariant?"*, or *"which data might be modified by a program?"*. Those questions are beyond the scope of the OCaml type system, but are within reach of abstract interpretation-based static analyses. The endeavour of supporting all the features of OCaml is beyond the scope of this project. Instead, we will target a representative subset of the pure fragment of the OCaml language, in which the core of Tezos's VM is written.

5 Social and environmental responsibility

5.1 Impact of research results

EPICURE has given rise to the INRIA exploratory action "Back to the trees" which aims to use probabilistic programming and Bayesian inference to produce a plant identification tool that is reliable, educational and convivial, built together with botanist collectives.

6 Highlights of the year

6.1 Awards

In 2023, Sandrine Blazy received:

- the ACM SIGPLAN Programming Languages Software award, with X. Leroy, Z. Dargaye, J.H. Jourdan, M.Schmidt, B. Schommer, and J.B. Tristan for the development of CompCert.
- the Lucas award from FME Board for a paper on the CompCert memory model, published at FM'06 with Z.Dargaye and X.Leroy,
- the CNRS Silver Medal.

7 New software, platforms, open data

7.1 New software

7.1.1 necro

Name: necro

Keywords: Semantics, Programming language, Specification language

Functional Description: The goal of the project is to provide a tool to manipulate skeletal semantics, a format to represent the semantics of programming languages. This tool has been mostly developed by Victoire Noizet.

URL: `http://skeletons.inria.fr/necro.html`

Publication: tel-03855276v1

Contact: Alan Schmitt

Participant: Alan Schmitt

7.1.2 Timbuk

Keywords: Automated deduction, Ocaml, Program verification, Tree Automata, Term Rewriting Systems

Functional Description: Timbuk is a tool designed to compute or over-approximate sets of terms reachable by a given term rewriting system. The libray also provides an OCaml toplevel with all usual functions on Bottom-up Nondeterministic Tree Automata.

URL: http://people.irisa.fr/Thomas.Genet/timbuk/index.html

Contact: Thomas Genet

Participant: Thomas Genet

7.1.3 dmap

Name: dependent maps library in OCaml

Keywords: Ocaml, Library, Data structures

Functional Description: dmap is an OCaml library that implements immutable maps, for which the type of data may depend on the key they are associated with.

URL: https://gitlab.inria.fr/bmontagu/dmap

Contact: Benoit Montagu

Participant: Benoit Montagu

7.1.4 sexp_decode

Keywords: Ocaml, Library

Functional Description: sexp_decode is an OCaml library of monadic combinators for decoding S-expressions (as defined in the Csexp library) into structured data.

URL: https://gitlab.inria.fr/bmontagu/sexp_decode

Contact: Benoit Montagu

Participant: Benoit Montagu

7.1.5 CompcertSSA

Keywords: Optimizing compiler, Formal methods, Proof assistant, SSA

Functional Description: CompcertSSA is built on top of the Compcert verified C compiler, by adding a middle-end based on the SSA form (Static Single Assignment) : conversion to SSA, SSA-based optimizations, and destruction of SSA.

URL: https://compcertssa.gitlabpages.inria.fr/

Publications: hal-01378393, hal-01193281, hal-02904204, hal-03899435, hal-01110783, hal-01097677, hal-01110779

Contact: Delphine Demange

Participants: Sandrine Blazy, Delphine Demange, Yon Fernandez de Retana, David Pichardie, Leo Stefanesco

7.2 Open data

8 New results

8.1 Skeletal Semantics

Participants: Martin Andrieux, , Thomas Jensen, , Victoire Noizet, , Vincent Rébiscoul, , Alan Schmitt.

The work on skeletal semantics [33], a modular and formal way to describe semantics or programming languages, has continued during 2023. Links to papers and tools can be found at the dedicated website. Several interns and PhD students are also working on skeletal semantics.

Victoire Noizet is the main designer of Skel, the skeletal semantics language, and the main developer of Necro, a tool to manipulate skeletal semantics. Victoire continued her work on the development and maintenance of Necro as an engineer. She presented her work on NecroML at JFLA 2023 [25].

Vincent Rébiscoul has continued working on static analyses for skeletal semantics. He is designing a framework that can automatically derive a control-flow analysis from the definition of a language as a skeletal semantics. The goal of the approach is to automatically derive the correctness of the analysis from the correctness of its components. His work has been published at Express/SOS 2023 [20].

Martin Andrieux is a M1 student doing his research project on a skeletal semantics of Python, based on the formal semantics written by Raphaël Monat [32]. His work has been accepted for presentation at the forthcoming conference JFLA 2024.

8.2 Static Analysis of Functional Programs

Participants: Thomas Genet, Thomas Jensen, Pierre Lermusiaux, Benoit Montagu.

The Salto project aims at developing a static analyser for OCaml programs based on abstract interpretation. A primary goal is to detect possibly uncaught exceptions in OCaml programs.

In 2023, the Salto prototype analyser reached a state where it is able to take real OCaml programs as input. A large subset of the OCaml language is now supported, which permits the analysis of non-trivial programs, up to a few thousand lines of code. As a next step, we want to cover more base types and more primitive functions, more features such as arrays and laziness, as well as more advanced features (recursive modules, objects,...).

Preliminary results of the Salto analyser have been presented [26] at the ML family workshop 2023.

A research article [31] has been accepted for publication at ESOP 2024,that describes the theory underlying the Salto static analyser, and that presents some experimental results.

8.3 Verification of Functional Programs using on Tree Automata

Participants: Théo Losekoot, Thomas Genet, Thomas Jensen.

We develop a specific theory and the related tools for analyzing functional programs manipulating algebraic data types. The domain and the co-domain of such functions are (generally) infinite set of terms. We use tree automata to finitely represent such infinite sets of terms. We have already shown how to exploit those informations using a *dedicated type system* associating regular language types to variables, expressions, etc. of a program. By automatically inferring such types we perform fully automatic verification of safety properties of tree-processing higher-order functional programs. Experiments are detailed here. Such regular abstractions are powerful but cannot represent relations between the input and the output of a function. This has been improved in [21], where we use *convoluted tree automata* to finitely approximate the infinite input-output relation of first-order functions manipulating algebraic data types. Experiments can be found here.

8.4 Machine checked proof of an rBPF virtual machine

Participants: Frédéric Besson, Shenghao Yuan, Benjamin Lion, Jean-Pierre Talpin.

The rBPF virtual machine adapts the eBPF (extended Berkeley Packet Filters) technology to resource constrained devices running the RIOT micro-kernel. Typically, eBPF programs are untrusted user-provided programs that are used to monitor the kernel behaviour.

As the VM runs with kernel privileges on micro-controllers which rarely feature hardware memory protection, isolation is an essential property that is needed to ensure system integrity against potentially malicious programs.

In previous works, we have shown how to derive, within the Coq proof assistant, the verified C implementation of an eBPF virtual machine from a Gallina specification [34]. We have streamlined the proof methodology and improved the performance of the VM using a cache mechanism speeding up dynamic memory checks [23].

8.5 Support for Cryptographic Constant-time Programming

Participants: Frédéric Besson, Thomas Jensen, Gautier Raimondi, Jean-Loup Hatchikian Houdot.

Cryptographic constant-time is a programming discipline for protecting against timing attacks. This discipline forbids branching or performing memory accesses depending on secrets.

We have developed a program transformation to enforce the constant-time property [16]. The transformation is based on an extension of the usual Volpano-Smith type-system that is flow-sensitive and which tracks precisely the conditionals responsible for indirect information flows. The transformation is directed by these more precise information flow properties. The main insight is that indirect information flows may be eliminated by performing an enhanced *if-conversion* which also transforms the continuation program. We are also investigating hardware support to get constant time memory accesses, in order to reduce the need for some of the most costly transformations [18].

8.6 Verified Compilation

Participants: Sandrine Blazy, Aurèle Barrière, Delphine Demange.

In 2023, we continued our work on verified compilation, focusing on the one hand on Just-in-Time compilers (or JIT) and on the other hand on extending the CompCertSSA compiler with a Gated Static Single Assignment (GSA) form.

New results on JIT compilation [14] concern the development of a JIT prototype with dynamic generation of native code, implemented and formally verified in Coq. Although some parts of a JIT cannot be written in Coq, we propose a proof methodology, first to ensure JIT correctness, and second to delimit, specify and reason on the impure effects of a JIT. We argue that the daunting task of formally verifying a complete JIT should draw on existing proofs of native code generation. To this end, our work successfully reuses CompCert and its correctness proofs during dynamic compilation. Finally, our prototype can be extracted and executed. This is a joint work with David Pichardie (Meta).

New results on GSA [19] concern the development of a prototype that adds the GSA intermediate representation to the CompCertSSA compiler, and a translation from SSA to GSA that is semantics preserving. This is a joint work with Yann Herklotz (PhD student from Imperial College London).

8.7 Multi-Token Geometry of Interaction and its Causal Unfolding

Participants: Simon Castellan.

The Geometry of Interaction (GoI) is a semantic framework that can unify operational and denotational semantics of higher-order programs. It views open programs as certain automata exchanging tokens with its environment. As it stands between operational semantics and denotational semantics it can be used to transfer results in between : to compute efficiently denotational semantics, or to reason compositionally via an operational semantics. Moreover, its automata presentation could make it possible to use automata-theoeric results in order to verify properties of programs.

Unfortunately, traditional GoI is well-studied only for pure functional programs. There has been extensions to accomodate various effects but through the use of ad-hoc models that make it difficult to reap the benefits of the approach.

In [15], we provide a Geometry of Interaction for concurrent programs, making a bridge between the traditional operational semantics of a call-by-name shared-memory language, and its denotational semantics in terms of event structures (a causal model of concurrency based on partial orders). Instead of ad-hoc automata, we use *coloured open Petri nets* to represent programs. We show how to represent programs as Petri nets, and how to unfold the Petri nets into event structures. This allows us to show a strong correspondance result between the operational semantics and the denotational semantics, beyond what was already proved.

This opens the way of transferring algorithms on Petri nets to do static analysis on concurrent programs.

This is joint work with Pierre Clairambault.

8.8 WebSpec: Towards Machine-Checked Analysis of Browser Security Mechanisms

Participants: Benjamin Farinier.

Web browsers are considered among the most complex software in use today, and the number of Web platform components is constantly increasing. These components, which include browser functionalities and security mechanisms, are introduced as W3C Editor's Drafts and undergo manual expert reviews before becoming W3C recommendations. However, these manual reviews often overlook logical flaws, leading to critical security vulnerabilities in Web technologies. This situation stems from several concurring factors:

- Web platform components are typically specified informally, making it difficult to identify potential corner cases and vulnerabilities even during expert reviews.

- There is no precise understanding of which security properties should be considered invariant in the Web.

WEBSPEC [22] is a formal framework for the security analysis of browser security mechanisms. It is designed to move away from manual expert reviews and towards a more formal and automated approach to identifying security flaws and vulnerabilities. WEBSPEC includes a formal browser model written in Coq. This model covers a core set of Web platform components, both well-established (such as cookies, Same-Origin Policy, CORS) and recent ones (like CSP level 3 and Trusted Types). The WEBSPEC toolchain also comprises a compiler and a trace verifier. The compiler translates the browser model and Web invariants into SMT formulas, enabling model checking using the Z3 automated theorem prover. This toolchain can be used for both identifying security bugs and generating machine-checked security proofs.

Joint work with Lorenzo Veronese, Pedro Bernardo, Mauro Tempesta, Marco Squarcina, Matteo Maffei.

9 Bilateral contracts and grants with industry

9.1 Bilateral contracts with industry

Salto: static analyses for OCaml programs

Participants: Thomas Genet, Thomas Jensen, Pierre Lermusiaux, Benoît Montagu.

Title: Salto

Industrial partner: Nomadic Labs

Date/Duration: two years (Nov 2022 – Oct 2024)

Participants: Thomas Genet, Thomas Jensen, Pierre Lermusiaux, Benoît Montagu

Additional info/keywords: As part of the Inria-Nomadic Labs partnership, the EPICURE research team is working on the development of Salto, a static analyzer for OCaml programs. The goal of this analyzer is to help the Nomadic Labs engineers improve the trust on their OCaml code-base, that implements the runtime system for the Tezos blockchain. The Salto static analyzer builds upon abstract interpretation techniques and recent work on control-flow analyses [13] and regular tree languages [10] that are developed in our research team. The aim of the Salto static analyzer is to detect whether an OCaml program might violate some safety properties, such as: May a program raise some uncaught exception? May a program violate some user-defined assertion or invariant? May a program access some data outside the bounds of an array? May a program perform some undesired arithmetic overflow?

10 Partnerships and cooperations

10.1 International research visitors

10.1.1 Visits of international scientists

Other international visits to the team

Christine Rizkallah

Status Senior Lecturer

Institution of origin: University of Melbourne

Country: Australia

Dates: May 2nd – May 5th

Context of the visit: invitation to give a research talk to the team's seminar

10.2 European initiatives

PHC Polonium

Participants: Alan Schmitt.

Title: Polonium

Partner Institution(s): University of Wrocław, Poland

Date/Duration: one year

Additional info/keywords: Alan Schmitt is part of a Polonium Hubert Curien Partnership (PHC) with the University of Wrocław. This partnership is led by Sergueï Lenglet, from Loria, Nancy. We work with Małgorzata Biernacka and Dariusz Biernacki on formal transformations of operational semantics.

10.3 National initiatives

10.3.1 The ANR CISC project

Participants: Thomas Jensen, Victoire Noizet, Alan Schmitt.

The goal of the CISC project is to investigate multitier languages and compilers to build secure IoT applications with private communication. In particular, we aim at extending multitier platforms by a new orchestration language that we call Hiphop.js to synchronize internal and external activities of IoT applications as a whole. Our goal is to define language, semantics, attacker models, and policies for the IoT and investigate automatic implementation of privacy and security policies by multitier compilation of IoT applications. To guarantee such applications are correct, and in particular that the required security and privacy properties are achieved, we propose to certify them using the Coq proof assistant. We plan to implement the CISC results as extensions of the multitier language Hop.js (developed at Inria) to maximize its impact. Using the new platform, we will carry out experimental studies on IoT security.

The project partners include the following Inria teams: Épicure, Collège de France, Indes, and Privatics. The project has from April 2018 to September 2023.

10.3.2 ANR SCRYPT

Participants: Thomas Jensen, Frederic Besson, Gautier Raimondi, Jean-Loup Hatchikian-Houdot.

The Scrypt project (ANR-18-CE25-0014) aims at providing secure implementations of cryptographic primitives using formal methods and secure compilation techniques. One specific goal is to design secure compilers which preserve the security of the source code against side-channel attacks.

This is a joint project with the Inria team Marelle, École Polytechnique and the company AMOSSYS.

10.3.3 PEPR Cybersécurité Secureval

Participants: Thomas Jensen, Frederic Besson, Sandrine Blazy, Benjamin Farinier, Alexandre Drewery, Clement Chavanon.

The Secureval project concerns the assessment of the security of digital systems. Digital system security assessment relies on compliance and vulnerability analyses to provide recognized cybersecurity assurances. Innovative tools will be designed around new digital technologies in order to verify the absence of hardware and software vulnerabilities, and to carry out the required proof of conformity. EPICURE contributes with research on advanced static analysis and verified compilation techniques.

10.4 Regional initiatives

10.4.1 Labex Combinlabs SCRATCHS project

Participants: Frederic Besson, Jean-Loup Hatchikian-Houdot.

The goal of the SCRATCHS project (2021-2024) is to co-design a compiler toolchain and a RISC-V micro-controller in order to ensure the absence of side-channel timing leaks. The contribution of the EPICURE team is focused on how to exploit the security mechanisms offered by the processor in a dedicated secure compiler toolchain. SCRATCHS is a joint project between the EPICURE team, the INRIA SUSHI team and the Lab-Sticc ARCAD team, and is funded by the Laboratoire d'excellence Combinlabs.

11 Dissemination

11.1 Promoting scientific activities

Participants: Delphine Demange, Sandrine Blazy, Thomas Genet, Thomas Jensen, Frédéric Besson, Benoît Montagu, Alan Schmitt.

11.1.1 Scientific events: organisation

- Alan Schmitt: Steering Committee of JFLA

General chair, scientific chair

- Delphine Demange: General chair for JFLA 2024

Member of the organizing committees

- Delphine Demange: Organizing committee for JFLA 2023, and JFLA 2024.

11.1.2 Scientific events: selection

Member of the conference program committees

- Benoît Montagu: Symposium on Implementation and Application of Functional Languages (IFL'23)
- Benoît Montagu: ML family workshop 2023
- Benoît Montagu: OCaml workshop 2023
- Delphine Demange: JFLA 2023
- Delphine Demange: CGO Student Research Competition
- Sandrine Blazy: Conference on Interactive Theorem Proving ITP'23
- Sandrine Blazy: Symposium on FOrmal Methodes FM'23
- Sandrine Blazy: Workshop on Teaching Formal Methods FMTea'23
- Sandrine Blazy: European Symposium on Programming ESOP'23
- Sandrine Blazy: AFADL'23

11.1.3 Journal

Member of the editorial boards Sandrine Blazy is a member of the editorial board of the LMCS journal.

11.1.4 Invited talks

- “Formalizing Real World Programming Languages with Skeletal Semantics”, Alan Schmitt, Express/SOS 2023
- “Software Security: leave it to the compiler”, Frédéric Besson, Forum International de la Cybersécurité (FIC 2023)
- "CompCert: a journey through the landscape of mechanized semantics for verified compilation", Sandrine Blazy, ACM Conference on Certified Programs and Proofs (CPP’23)
- "How to provide proof that software is bug-free? Verified compilation to the rescue", Sandrine Blazy, National days of GDR GPL

11.1.5 Leadership within the scientific community

- Sandrine Blazy was President of the jury for the open science prize for free research software, organized by the French Ministry of Higher Education and Research (8 prizes were awarded)
- Sandrine Blazy is a member of the ACM SIGPLAN committee for the Robin Milner Young Researcher award

11.1.6 Scientific expertise

- Alan Schmitt, Member of Conseil Scientifique of LMF, Formal Methods Laboratory, Paris Saclay
- Thomas Jensen was reviewer for the ERC Advanced Grants selection.

11.1.7 Research administration

- Sandrine Blazy is deputy director of the IRISA CNRS laboratory.
- Thomas Jensen is director of the Laboratoire d’excellence CominLabs.

11.2 Teaching - Supervision - Juries

Participants: Delphine Demange, Sandrine Blazy, Thomas Genet, Thomas Jensen, Frédéric Besson, Benoît Montagu, Alan Schmitt, Simon Castellan, Benjamin Farinier.

11.2.1 Teaching

- Master : Alan Schmitt, Advanced Semantics, 30h, M2, ENS Rennes, France
- Master : Alan Schmitt, Preparation of Agregation exam, 62h, M2, ENS Rennes, France
- Licence : Benoît Montagu, Programmation de Confiance, 36h, L3, Université Rennes, France
- Master : Benoît Montagu, Analyse et Conception Formelles, 24h, M1, Université Rennes, France
- Licence : Frédéric Besson, Programmation Fonctionnelle, 28h, L3, Insa, France
- Licence : Delphine Demange, Programmation Impérative, 55h, L1, Université de Rennes, France
- Licence : Delphine Demange, Algorithmique et Complexité, 40h, L1, Université de Rennes, France
- Licence : Sandrine Blazy, Programmation de Confiance, 55h, L3, Université Rennes, France
- Master : Sandrine Blazy, Mechanized Semantics, 32h, M1, Université Rennes, France

- Doctorate : Sandrine Blazy, Compiler Verification, Oregon Programming Languages summer school, 4.5h, Eugene (USA)
- Doctorate : Sandrine Blazy, Compiler Verification, Verification Technology summer school, Systems & Applications, 6h, Nancy
- Master : Thomas Jensen, Software Security, 20 h, University of Rennes.

11.2.2 Supervision

- PhD in progress: Vincent Rébiscoul, "Analyses Statiques pour Sémantiques Squelettiques", since September 2020, Thomas Jensen and Alan Schmitt.
- PhD in progress: Alexandre Drewery, "Analyse statique incrémentale pour la sécurité logicielle", since September 2023, Thomas Jensen and David Pichardie.
- PhD in progress: Théo Losekoot, "Verification of relational properties with tree automata", since September 2021, Thomas Jensen and Thomas Genet.
- PhD in progress: Malo Revel, "Proving regular theorems on functional programs", since September 2023, Thomas Jensen and Thomas Genet.
- PhD in progress: Jean-Loup Hatchikian-Houdot, "Security-enhancing compiler against side-channel attacks", since Octobre 2021, Guillaume Hiet and Frédéric Besson.
- PhD in progress: Clément Chavanon, "Refinement of formal specifications for secure environements" since September 2023, Sandrine Blazy and Frédéric Besson.
- PhD in progress: Romeo La Spina, "Analyse de flot de données et de dépendances pour la compilation optimisante vérifiée", Sandrine Blazy and Delphine Demange.
- PhD in progress: Tony Law, "Formally verified high-level synthesis", Sandrine Blazy and Delphine Demange.
- PhD in progress: Alain Delaët-Tixieuil, "Verified compilation for a language describing the law", Sandrine Blazy and Denis Merigoux.
- PhD defended: Santiago Sara Bautista, "Static Analysis of Algebraic Data Types and Arrays" [29], supervised by Thomas Jensen and Benoît Montagu.
- PhD defended: Gautier Raimondi, "Secure compilation against side-channel attacks", Thomas Jensen and Frédéric Besson.

11.2.3 Juries

- Alan Schmitt, jury member (reviewer) for the PhD defense of Jayanth Krishnamurthy, April 2023, Université de Nice Côte d'Azur
- Alan Schmitt, jury member for the PhD defense of Loïc Germerie Guizouarn, December 2023, Université de Nice Côte d'Azur
- Sandrine Blazy, jury member for the HDR defense of Arthur Charguéraud, 02/2023, Strasbourg University
- Sandrine Blazy, jury member (reviewer) for the HDR defense of Claire Maiza, 6/2023, University Grenoble Alpes
- Sandrine Blazy, jury member (reviewer) for the PhD defense of Baptiste Pollien, 11/2023, University of Toulouse
- Sandrine Blazy, jury member (reviewer) for the PhD defense of Swarn Priya, 11/2023, Université de Nice Côte d'Azur

- Sandrine Blazy, jury member (president) for the PhD defense of Shenghao Yuan, 12/2023, University of Rennes
- Sandrine Blazy, jury member (president) for the PhD defense of Gautier Raimondi, 12/2023, University of Rennes
- Sandrine Blazy, jury member (reviewer) for the PhD defense of Enzo Crance, 12/2023, University of Nantes
- Thomas Jensen, jury member (president) for the PhD defence of Gwendal Patat, 12/2023, University of Rennes.
- Delphine Demange, jury member (examiner) for the PhD defense of Léo Gourdin, 12/2023, University of Grenoble Alpes.

11.3 Popularization

Participants: Sandrine Blazy, Thomas Genet.

11.3.1 Articles and contents

- Sandrine Blazy: "What you need to know about compilers and their verification", Binaire, Le Monde, 12/2023, part 2 and part 1

11.3.2 Interventions

- Thomas Genet: "Blockchain. What is it, how it works, can it be useful for something?", given in two High School close to Rennes.
- Thomas Genet: "Bug, virus, pirates. So many threats and no solution? Yes, mathematics.", given in one High School close to Rennes.

12 Scientific production

12.1 Major publications

[1] O. Andreescu, T. Jensen, S. Lescuyer and B. Montagu. ‘Inferring frame conditions with static correlation analysis’. In: *Proceedings of the ACM on Programming Languages* 3.POPL (2nd Jan. 2019), pp. 1–29. DOI: 10.1145/3290360. URL: https://hal.inria.fr/hal-02413262.

[2] A. Barrière, S. Blazy, O. Flückiger, D. Pichardie and J. Vitek. ‘Formally verified speculation and deoptimization in a JIT compiler’. In: *Proceedings of the ACM on Programming Languages* 5.POPL (4th Jan. 2021), p. 26. DOI: 10.1145/3434327. URL: https://hal.science/hal-03185848.

[3] A. Barrière, S. Blazy and D. Pichardie. ‘Formally Verified Native Code Generation in an Effectful JIT - or: Turning the CompCert Backend into a Formally Verified JIT Compiler’. In: *Proceedings of the ACM on Programming Languages* (Jan. 2023). DOI: 10.1145/3571202. URL: https://hal.inria.fr/hal-03882598.

[4] G. Barthe, S. Blazy, B. Grégoire, R. Hutin, V. Laporte, D. Pichardie and A. Trieu. ‘Formal verification of a constant-time preserving C compiler’. In: *Proceedings of the ACM on Programming Languages* 4.POPL (Jan. 2020), pp. 1–30. DOI: 10.1145/3371075. URL: https://hal.univ-lorraine.fr/hal-02975012.

[5] F. Besson, S. Blazy, A. Dang, T. Jensen and P. Wilke. ‘Compiling Sandboxes: Formally Verified Software Fault Isolation’. In: ESOP 2019 - 28th European Symposium on Programming. Vol. 11423. LNCS. Prague, Czech Republic: Springer, 6th Apr. 2019, pp. 499–524. DOI: 10.1007/978-3-030-17184-1_18. URL: https://hal.inria.fr/hal-02316189.

[6] F. Besson, A. Dang and T. Jensen. 'Information-Flow Preservation in Compiler Optimisations'. In: CSF 2019 - 32nd IEEE Computer Security Foundations Symposium. Hoboken, United States: IEEE, 25th June 2019, pp. 1–13. URL: https://hal.inria.fr/hal-02180303.

[7] M. Bodin, A. Charguéraud, D. Filaretti, P. Gardner, S. Maffeis, D. Naudziuniene, A. Schmitt and G. Smith. 'A Trusted Mechanised JavaScript Specification'. In: POPL 2014 - 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. San Diego, United States, 22nd Jan. 2014. URL: https://hal.inria.fr/hal-00910135.

[8] M. Bodin, P. Gardner, T. Jensen and A. Schmitt. 'Skeletal Semantics and their Interpretations'. In: *Proceedings of the ACM on Programming Languages* 44 (2019), pp. 1–31. DOI: 10.1145/3290357. URL: https://hal.inria.fr/hal-01881863.

[9] S. Castellan and P. Clairambault. 'The Geometry of Causality: Multi-Token Geometry of Interaction and its Causal Unfolding'. In: *Proceedings of the ACM on Programming Languages* (Jan. 2023). URL: https://hal.science/hal-03286443.

[10] T. Haudebourg, T. Genet and T. Jensen. 'Regular Language Type Inference with Term Rewriting - extended version'. In: *Proceedings of the ACM on Programming Languages.* International Conference on Functional Programming (ICFP) 4.ICFP (2020), pp. 1–29. DOI: 10.1145/3408994. URL: https://hal.inria.fr/hal-02795484.

[11] J.-H. Jourdan, V. Laporte, S. Blazy, X. Leroy and D. Pichardie. 'A formally-verified C static analyzer'. In: POPL 2015: 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. Mumbai, India: ACM, 15th Jan. 2015, pp. 247–259. DOI: 10.1145/2676726.2676966. URL: https://hal.inria.fr/hal-01078386.

[12] B. Montagu and T. Jensen. 'Stable relations and abstract interpretation of higher-order programs'. In: *Proceedings of the ACM on Programming Languages* 4.ICFP (2nd Aug. 2020), pp. 1–30. DOI: 10.1145/3409001. URL: https://hal.inria.fr/hal-02916996.

[13] B. Montagu and T. Jensen. 'Trace-Based Control-Flow Analysis'. In: PLDI 2021 - 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. Virtual, Canada: ACM, 20th June 2021, pp. 1–15. DOI: 10.1145/3453483.3454057. URL: https://hal.inria.fr/hal-03266981.

12.2 Publications of the year

International journals

[14] A. Barrière, S. Blazy and D. Pichardie. 'Formally Verified Native Code Generation in an Effectful JIT - or: Turning the CompCert Backend into a Formally Verified JIT Compiler'. In: *Proceedings of the ACM on Programming Languages* (Jan. 2023), pp. 1–28. DOI: 10.1145/3571202. URL: https://inria.hal.science/hal-03882598.

[15] S. Castellan and P. Clairambault. 'The Geometry of Causality: Multi-Token Geometry of Interaction and its Causal Unfolding'. In: *Proceedings of the ACM on Programming Languages* (Jan. 2023), pp. 1–70. URL: https://hal.science/hal-03286443.

International peer-reviewed conferences

[16] F. Besson, T. Jensen and G. Raimondi. 'Type-directed Program Transformation for Constant-Time Enforcement'. In: PPDP 2023 - International Symposium on Principles and Practice of Declarative Programming. Lisboa, Portugal: ACM, 22nd Oct. 2023, pp. 1–13. DOI: 10.1145/3610612.3610618. URL: https://inria.hal.science/hal-04268830.

[17] S. Castellan, J. Käfer and E. Tannier. 'Back to the trees: Identifying plants with Human Intelligence'. In: LIMITS 2023 - Ninth Workshop on Computing within Limits. Virtual, France: LIMITS, 14th June 2023, pp. 1–11. DOI: 10.21428/bf6fb269.265c52ce. URL: https://hal.science/hal-04121511.

[18] N. Gaudin, J.-L. Hatchikian-Houdot, F. Besson, P. Cotret, G. Guy, G. Hiet, V. Lapotre and P. Wilke. 'Work in Progress: Thwarting Timing Attacks in Microcontrollers using Fine-grained Hardware Protections'. In: *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). Delft, Netherlands, 3rd July 2023, pp. 1–7. URL: https://hal.science/hal-04155139.

[19] Y. Herklotz, D. Demange and S. Blazy. 'Mechanised Semantics for Gated Static Single Assignment'. In: CPP 2023 - 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. Boston, United States: ACM, 16th Jan. 2023. DOI: 10.1145/3573105.3575681. URL: https://inria.hal.science/hal-03899435.

[20] T. Jensen, V. Rébiscoul and A. Schmitt. 'Deriving Abstract Interpreters from Skeletal Semantics'. In: EXPRESS/SOS 2023 - 30th International Workshop on Expressiveness in Concurrency and 20th Workshop on Structural Operational Semantics. Vol. 387. Antwerp, Belgium, 12th Sept. 2023, pp. 97–113. DOI: 10.4204/EPTCS.387.8. URL: https://inria.hal.science/hal-04207565.

[21] T. Losekoot, T. Genet and T. Jensen. 'Automata-Based Verification of Relational Properties of Functions over Algebraic Data Structures'. In: *Lipics*. FSCD 2023 - 8th International Conference on Formal Structures for Computation and Deduction. Rome, Italy: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 1–21. DOI: 10.4230/LIPIcs.FSCD.2023.7. URL: https://inria.hal.science/hal-04216680.

[22] L. Veronese, B. Farinier, P. Bernardo, M. Tempesta, M. Squarcina and M. Maffei. 'WebSpec: Towards Machine-Checked Analysis of Browser Security Mechanisms'. In: SP 2023 - 44th IEEE Symposium on Security and Privacy. San Francisco, United States: IEEE, 2023, pp. 2761–2779. DOI: 10.1109/SP46215.2023.10179465. URL: https://inria.hal.science/hal-04316042.

[23] S. Yuan, B. Lion, F. Besson and J.-P. Talpin. 'Making an eBPF Virtual Machine Faster on Microcontrollers: Verified Optimization and Proof Simplification'. In: SETTA 2023 - 9th International Symposium Dependable Software Engineering. Theories, Tools, and Applications. Vol. 14464. Lecture Notes in Computer Science. Nanjing (Chine), China: Springer Nature Singapore, 30th Nov. 2023, pp. 385–401. DOI: 10.1007/978-981-99-8664-4_22. URL: https://inria.hal.science/hal-04376380.

National peer-reviewed Conferences

[24] T. Jensen, V. Rébiscoul and A. Schmitt. 'Building CFA for λ-calculus from Skeletal Semantics'. In: *Journées Francophones des Langages Applicatifs*. JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs. Praz-sur-Arly, France, 16th Jan. 2023, pp. 152–171. URL: https://inria.hal.science/hal-03936686.

[25] L. Noizet and A. Schmitt. 'Necro ML: Kit de Nécromancie: Démonstration'. In: *Journées Francophones des Langages Applicatifs*. JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs. Praz-sur-Arly, France, 16th Jan. 2023, pp. 296–298. URL: https://inria.hal.science/hal-03936885.

Conferences without proceedings

[26] B. Montagu. 'The Design and Implementation of an Abstract Interpreter for OCaml Programs: A Preliminary Report on the Salto Analyser'. In: ML Family 2023 - Higher-order, Typed, Inferred, Strict: ACM SIGPLAN ML Family Workshop. Seattle, Washington, United States, 2023, pp. 1–4. URL: https://inria.hal.science/hal-04259875.

Scientific book chapters

[27] S. Castellan, P. Clairambault and G. Winskel. 'The Mays and Musts of Concurrent Strategies'. In: *Samson Abramsky on Logic and Structure in Computer Science and Beyond*. Vol. 25. Outstanding Contributions to Logic. Springer International Publishing, 2nd Aug. 2023, pp. 327–361. DOI: 10.1007/978-3-031-24117-8_9. URL: https://hal.science/hal-04244682.

Edition (books, proceedings, special issue of a journal)

[28] T. Bourke and D. Demange, eds. *JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs*. JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs. Journées Francophones des Langages Applicatifs. 31st Jan. 2023, pp. 1–308. URL: https://inria.hal.science/hal-03962188.

Doctoral dissertations and habilitation theses

[29] S. Bautista. 'Static Analysis of Algebraic Data Types and Arrays'. ENS Rennes, 20th Dec. 2023. URL: https://hal.science/tel-04379086.

Reports & preprints

[30] S. Bautista, T. Jensen and B. Montagu. *Lifting Numeric Relational Domains to Algebraic Data Types (extended version)*. Centre Inria de l'Université de Rennes; Univ Rennes, 26th June 2023. URL: https://inria.hal.science/hal-03765357.

12.3 Cited publications

[31] P. Lermusiaux and B. Montagu. 'Detection of Uncaught Exceptions in Functional Programs by Abstract Interpretation'. In: *Proceedings of the European Symposium On Programming (ESOP 2024)*. To appear. 2024.

[32] R. Monat. 'Static Type and Value Analysis by Abstract Interpretation of Python Programs with Native C Libraries'. PhD Thesis. Sorbonne Université, 2021.

[33] L. Noizet and A. Schmitt. 'Semantics in Skel and Necro'. In: *ICTCS 2022 - Italian Conference on Theoretical Computer Science*. CEUR Workshop Proceedings. Rome, Italy, Sept. 2022, pp. 1–17. URL: https://inria.hal.science/hal-03784478.

[34] S. Yuan, F. Besson, J.-P. Talpin, S. Hym, K. Zandberg and E. Baccelli. 'End-to-end Mechanized Proof of an eBPF Virtual Machine for Micro-controllers'. In: *CAV 2022 - 34th International Conference on Computer Aided Verification*. Haifa, Israel, Aug. 2022, pp. 1–23. URL: https://inria.hal.science/hal-03888082.