UMR IRISA

Activity Report 2022

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

Université de Rennes | rennes | INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES | cnrs | Inria | Inserm | CentraleSupélec | IMT Atlantique Bretagne-Pays de la Loire École Mines-Télécom | Université Bretagne Sud ubs

Contents

Project-Team EPICURE 1

1 Team members, visitors, external collaborators 2

2 Overall objectives 3

3 Research program 3

4 Application domains 4
4.1 Internet of Things . . . 4
4.2 High-assurance blockchains . . . 4

5 Highlights of the year 5
5.1 Awards . . . 5

6 New software and platforms 5
6.1 New software . . . 5
6.1.1 necro . . . 5
6.1.2 Timbuk . . . 5
6.1.3 dmap . . . 5
6.1.4 sexp_decode . . . 6
6.1.5 CompcertSSA . . . 6

7 New results 6
7.1 Skeletal Semantics . . . 6
7.2 Input-Output Relational Analyses for Functional Programs . . . 7
7.3 Machine checked proof of an rBPF virtual machine . . . 7
7.4 Mechanised Semantics for Gated Static Single Assignment . . . 7
7.5 Formal verification of JIT compilation . . . 8
7.6 Multi-Token Geometry of Interaction and its Causal Unfolding . . . 8

8 Bilateral contracts and grants with industry 9
8.1 Bilateral contracts with industry . . . 9

9 Partnerships and cooperations 9
9.1 International initiatives . . . 9
9.1.1 Participation in other International Programs . . . 9
9.2 European initiatives . . . 10
9.2.1 H2020 project: SPARTA Cybersecurity Competence Network . . . 10
9.3 National initiatives . . . 10
9.3.1 The ANR CISC project . . . 10
9.3.2 The ANR SCRYPT project . . . 11
9.4 Regional initiatives . . . 11
9.4.1 Labex Combinlabs SCRATCHS project . . . 11

10 Dissemination 11
10.1 Promoting scientific activities . . . 11
10.1.1 Scientific events: organisation . . . 11
10.1.2 Scientific events: selection . . . 12
10.1.3 Journal . . . 12
10.1.4 Invited talks . . . 12
10.1.5 Scientific expertise . . . 13
10.1.6 Research administration . . . 13
10.2 Teaching - Supervision - Juries . . . 13
10.2.1 Supervision . . . 13

10.2.2 Juries 14
10.3 Popularization 15
10.3.1 Internal or external Inria responsibilities 15
10.3.2 Education 15

11 Scientific production **15**
11.1 Major publications 15
11.2 Publications of the year 16
11.3 Cited publications 18

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]
- Frederic Besson [INRIA, Researcher]
- Simon Castellan [INRIA, Researcher]
- Benoit Montagu [INRIA, Researcher]
- Alan Schmitt [INRIA, Senior Researcher, HDR]

Faculty Members

- Sandrine Blazy [UNIV RENNES I, Professor, HDR]
- Delphine Demange [UNIV RENNES I, Associate Professor]
- Benjamin Farinier [UNIV RENNES I, Associate Professor, from Sep 2022]
- Thomas Genet [UNIV RENNES I, Professor, HDR]
- Remi Hutin [ENS RENNES, ATER]

PhD Students

- Aurele Barriere [ENS RENNES]
- Santiago Bautista [ENS RENNES]
- Jean-Loup Hatchikian-Houdot [INRIA]
- Adam Khayam [INRIA]
- Roméo La Spina [UNIV RENNES I, from Sep 2022]
- Tony Law [UNIV RENNES I, from Oct 2022]
- Gautier Raimondi [INRIA]

Technical Staff

- Pierre Lermusiaux [Inria, Engineer, from Nov 2022]
- Louis Noizet [INRIA, Engineer]

Interns and Apprentices

- Romeo La Spina [ENS RENNES, from Mar 2022 until Jul 2022]

Administrative Assistant

- Stephanie Gosselin Lemaile [INRIA]

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. We thus want to investigate how our techniques can be brought to bear on multitier programming languages. 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 propose 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 Highlights of the year

5.1 Awards

- In June 2022, Sandrine Blazy received ACM Software System Award, with X. Leroy, Z. Dargaye, J.H. Jourdan, M. Schmidt, B. Schommer, and J.B. Tristan for the development of the CompCert verified compiler.

6 New software and platforms

6.1 New software

6.1.1 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.

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

Contact: Alan Schmitt

6.1.2 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

6.1.3 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

6.1.4 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

6.1.5 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 New results

7.1 Skeletal Semantics

Participants: Guillaume Ambal, Martin Andrieux, Thomas Jensen, Adam Khayam, Louis Noizet, Vincent Rébiscoul, Alan Schmitt .

The work on skeletal semantics [22], a modular and formal way to describe semantics or programming languages, has continued during 2022. Links to papers and tools can be found at the dedicated website. Several interns and PhD students are also working on skeletal semantics.

Louis Noizet is the main designer of Skel, the skeletal semantics language, and the main developer of Necro, a tool to manipulate skeletal semantics. Louis successfully defended his PhD thesis in September 2022. He has now been hired as an engineer to continue the development and maintenance of Necro.

Guillaume Ambal is studying the inter-derivation of multiple semantics from a given language written in Skel. He has formalized in Coq a meta-semantics for skeletal programs that corresponds to an abstract machine. This meta-semantics describes in Coq how to evaluate a skeletal semantics in a step by step way. It is proven correct with respect to the usual big-step meta-semantics (i.e., a natural semantics for Skel), and an OCaml interpreter can be extracted from it. This thus provides a way of creating a certified interpreter for any arbitrary language expressed in Skel [16]. In addition, Guillaume had developed a tool that generates a small step semantics from a big step semantics. The tool also generates a Coq proof of the equivalence of the semantics [17]. Guillaume successfully defended his PhD in October 2022, he is now a postdoc at Imperial College.

Adam Khayam continued working on the formal semantics of JavaScript in Skel [21], in particular on the techniques required to faithfully model the delimited computations that are used in the standard. He has also worked on the semantics of Webi, a language describing web services and IOT devices. He has shown that using a special scheduler can significantly lessen the non-determinism of computations while retaining the same expressiveness. Adam has successfully defended his PhD in November 2022. He has accepted a postdoc position at INRIA Paris to start in January 2023.

Vincent Rébiscoul is 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. A preliminary version of his work has been accepted to be presented at JFLA 2023.

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 [31]. The first challenge to overcome has been the precise description of how scopes are handled in Python.

7.2 Input-Output Relational Analyses for Functional Programs

Participants: Santiago Bautista, Thomas Jensen, Benoît Montagu.

The goal of input-output relational static analyses—also known as *transformation* analyses—is to automatically infer specifications for a program's behaviors, by means of a mathematical relation that relates the inputs of a program to the outputs it may produce. We have shown in previous work [28] that such relations might provide great help to proof engineers, that work on formal verification projects. Indeed, these relations can help discharge a large proportion of invariant preservation lemmas.

After promising results [28] for first-order functional programs and higher-order programs [32], we have focused on inferring input-output relations for functional programs that manipulate *both* algebraic data-types *and* perform arithmetic operations at the same time. For this purpose, Santiago Bautista designed RAND, an abstract domain that is parameterized over an arbitrary relational numeric abstract domain, and can express precise input-output relations for such programs. The RAND abstract domain is exploited to analyze programs written in a first-order imperative programming language, that features immutable data-structures. Our analyzer successfully infers precise input-output summaries for such programs, and in particular for examples adapted from the abstract specification of the seL4 verified operating system [30]. This new result was presented at SAS'22 [18].

7.3 Machine checked proof of an rBPF virtual machine

Participants: Frédéric Besson, Shenghao Yuan, Jean-Pierre Talpin, Samuel Hym, Koen Zandberg, Emmanuel Baccelli.

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.

We have shown how to directly derive, within the Coq proof assistant, the verified C implementation of an eBPF virtual machine from a Gallina specification [23]. Leveraging the formal semantics of the CompCert C compiler, we obtain an end-to-end theorem stating that the C code of our VM inherits the safety and security properties of the Gallina specification. Our refinement methodology ensures that the isolation property of the specification holds in the verified C implementation. Preliminary experiments demonstrate satisfying performance [24].

7.4 Mechanised Semantics for Gated Static Single Assignment

Participants: Sandrine Blazy, Delphine Demange.

The Gated Static Single Assignment (GSA) form was proposed by Ottenstein et al. in 1990, as an intermediate representation for implementing advanced static analyses and optimisation passes in compilers. Compared to SSA, GSA records additional data dependencies and provides more context, making optimisations more effective and allowing one to reason about programs as data-flow graphs.

Many practical implementations have been proposed that follow, more or less faithfully, Ottenstein et al.'s seminal paper. But many discrepancies remain between these, depending on the kind of dependencies they are supposed to track and to leverage in analyses and code optimisations.

In this work, we define a formal semantics for GSA, mechanised in Coq. In particular, we clarify the nature and the purpose of gates in GSA, and define control-flow insensitive semantics for them. We provide a specification that can be used as a reference description for GSA. We also specify a translation from SSA to GSA and prove that this specification is semantics-preserving. We demonstrate that the approach is practical by implementing the specification as a validated translation within the CompCertSSA verified compiler.

This is joint work with Yann Herklotz, PhD student at Imperial College London [20].

7.5 Formal verification of JIT compilation

Participants: Sandrine Blazy, Aurèle Barrière.

Modern Just-in-Time compilers (or JIT) typically interleave several mechanisms to execute a program. For faster startup times and to observe the initial behavior of an execution, interpretation can be initially used. But after a while, JITs dynamically produce native code for parts of the program they execute often. JIT routinely generate code under assumptions that may be invalidated at run-time, this allows for specialization of program code to the common case in order to avoid unnecessary overheads due to uncommon cases. This form of software speculation requires support for deoptimization when some of the assumptions fail to hold.

Moreover, although some time is spent compiling dynamically, this mechanism makes for much faster times for the remaining of the program execution. Such compilers are complex pieces of software with various components, and greatly rely on a precise interplay between the different languages being executed, including on-stack-replacement. Traditional static compilers like CompCert have been mechanized in proof assistants, but JITs have been scarcely formalized so far, partly due to their impure nature and their numerous components.

We have modeled 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 methodologies, 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 work was presented at POPL 2021 in January 2022 and at POPL 2023 in January 2023 [14].

This is joint work with David Pichardie (Meta).

7.6 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 Bilateral contracts and grants with industry

8.1 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

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

Additional info/keywords: As part of the Inria-Nomadic Labs partnership, the ÉPICURE 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 [33] and regular tree languages [29] 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?

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Participation in other International Programs

Polonium

Participants: Alan Schmitt, Guillaume Ambal.

Title: Polonium

Partner Institution(s): University of Wrocław, Poland

Date/Duration: one year

Additionnal 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.

CoVARSH

Participants: Sandrine Blazy, Delphine Demange, Tony Law.

Title: CoVARSH

Partner Institution(s): Imperial College London, Great Britain

Date/Duration: one year

Additionnal info/keywords: Sandrine Blazy received a funding from "Appel Unique" of INS2I (CNRS) for starting a collaboration with John Wickerson and his PhD student Yann Herklotz. We work on designing a new backend of the CompCert compiler, targeting high-level synthesis. We went on two visits to London for a week each, and Yann Herklotz came to spend a week in Rennes. This resulted in a paper that will be presented at the CPP conference in January 2023.

9.2 European initiatives

9.2.1 H2020 project: SPARTA Cybersecurity Competence Network

Participants: Frédéric Besson, Thomas Jensen.

SPARTA is a Cybersecurity Competence Network, supported by the EU's H2020 program, with the objective to develop and implement top-tier research and innovation collaborative actions. Guided by concrete challenges forming an ambitious Cybersecurity Research & Innovation Roadmap, SPARTA has set up unique collaboration means, leading the way in building transformative capabilities and forming a world-leading Cybersecurity Competence Network across the EU. The SPARTA consortium assembles 44 actors from 14 EU Member States at the intersection of scientific excellence, technological innovation, and societal sciences in cybersecurity.

We coordinated the participation of INRIA in the EU network pilot on Cybersecurity. The INRIA participation is focused on all aspects of IoT security, ranging from system level (the RIOT OS and secure networking) to secure orchestration and the protection of privacy of IoT applications. INRIA's participation in the network involves nine INRIA project-teams from the centres of Rennes (Celtique, Cidre,LHS), Paris (Eva) , Saclay (Grace, Infine), Sophia-Antipolis (Indes) Nancy (LHS) and Grenoble/Lyon (Privatics).

Thomas Jensen was editor of the Scientific Roadmap of Sparta [27].

9.3 National initiatives

9.3.1 The ANR CISC project

Participants: Thomas Jensen, Louis 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), based on the JavaScript language 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: Celtique, Collège de France, Indes, and Privatics. The project runs from April 2018 to September 2023.

9.3.2 The ANR SCRYPT project

Participants: Thomas Jensen, Frédéric 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.

9.4 Regional initiatives

9.4.1 Labex Combinlabs SCRATCHS project

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

The goal of SCRATCHS (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 CELTIQUE team will work at exploiting the security mechanisms in a dedicated secure compiler toolchain.

SCRATCHS is a joint project between the CELTTIQUE team, the CIDRE team and the Lab-Sticc ARCAD team

10 Dissemination

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

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

- Delphine Demange: Steering Committee of CC (2021-2024)
- Alan Schmitt: Steering Committee of JFLA
- Thomas Jensen, Steering Committee SAS.

General chair, scientific chair

- Delphine Demange: General co-chair of JFLA 2023

Member of the organizing committees

10.1.2 Scientific events: selection

Chair of conference program committees

- Benoît Montagu: Chair of the 2022 ML family workshop (affiliated with ICFP'22), Ljubljana, Slovenia

Member of the conference program committees

- Sandrine Blazy: FMTea 2023, ESOP 2023, FM 2023, AFADL 2022, Types 2022, ECOOP 2022, ITP 2022, JFLA 2022, POPL 2022
- Delphine Demange: CC'22, CGO'22, OOPSLA'22
- Benoît Montagu: OCaml workshop 2022
- Frédéric Besson: C&ESAR'22

Reviewer

- Alan Schmitt: CONCUR'22, PPDP'22
- Frédéric Besson: ITP'22

10.1.3 Journal

Member of the editorial boards

Reviewer - reviewing activities

- Alan Schmitt: TCS

10.1.4 Invited talks

- Sandrine Blazy, "Semantic preservation of constant-time policies during compilation", invited talk at the annual meeting of GT Formal Methods and Security of GDR Security, March 2022, Fréjus.
- Sandrine Blazy, "Obfuscation du logiciel : brouiller le code pour protéger les programmes", invited talk at College de France, course of Xavier Leroy on software security, April 2022, Paris.
- Sandrine Blazy, "CompCert: a journey through the landscape of verified compilation", invited talk at the annual meeting of GT VERIF of GDR IM, July 2022, Bordeaux.
- Sandrine Blazy, "Verified compilation: towards zero defect software", invited talk at the conference «Sciences du logiciel: de l'idée au binaire» organized by INS2I of CNRS, September 2022, Paris.
- Sandrine Blazy, invited speaker at the panel on "the challenges of compilation",Inria scientific days, November 2022, Rocquencourt.
- Sandrine Blazy, "The CompCert formally verified compiler", Imperial College London, guest lecture of the Software and Hardware verification course, December 2022.
- Delphine Demange, "Si2-FIP: Programmation Fonctionnelle en Licence 1 avec Scala", invited talk at JFLA 2022.
- Thomas Genet, "Using Regular Tree Languages for Verification, at last", invited talk at FROM 2022.

- Benoît Montagu, “Formal Verification of an Industrial Micro-Kernel: An Experience Report”, ENS Rennes seminar
- Benoît Montagu, “Trace-Based Control-Flow Analysis”, POPV seminar, Boston University
- Benoît Montagu, “Salto: Static Analyses for Trustworthy OCaml”, Inria/Nomadic Labs Scientific Days
- Benoît Montagu, “Stable Relations and Abstract Interpretation of Higher-Order Programs”, session for in-person talks at OOPSLA’22 for COVID-time ICFP’20 papers
- Benoît Montagu, “Formal Verification of an Industrial Micro-Kernel: An Experience Report”, University of Melbourne seminar
- Thomas Jensen, “Formal methods for software security”, Digital Tech Summit, Copenhagen, Denmark.

10.1.5 Scientific expertise

- Sandrine Blazy: member of the ACM SIGPLAN committee for the Robin Milner Young Researcher Award.
- Sandrine Blazy: member of the scientific committee of the Cybersecurity program (Flanders, Belgium).

10.1.6 Research administration

- Sandrine Blazy is deputy director of IRISA UMR 6074 since January 2021.
- Sandrine Blazy is a member of the scientific committee of GDR GPL of CNRS.
- Thomas Jensen is director of the LabEx CominLabs since July 2022.

10.2 Teaching - Supervision - Juries

10.2.1 Supervision

- PhD in progress: Alain Delaët-TIXEUIL, "Interactive proof of programs derived from legislative specifications", since October 2022, Sandrine Blazy and Denis Merigoux (team Prosecco).
- PhD in progress: Roméo La Spina, "Analyse de Flot de Données et Dépendences pour la Compilation Optimisante Vérifiée", since September 2022, Sandrine Blazy and Delphine Demange.
- PhD in progress: Tony Law, "Efficient, Correct and Practical High-Level Synthesis", since October 2022, Sandrine Blazy and Delphine Demange.
- Master 2 Internship: Roméo La Spina, "Un analyseur flots de données paramétré par un ordre d’itération pour le compilateur formellement vérifié CompCert", February–July 2022, Sandrine Blazy and Delphine Demange.
- PhD in progress: Santiago Bautista, “Static analyses for semi-interactive program verification”, since September 2020, Thomas Jensen and Benoît Montagu.
- PhD in progress: Théo Losekoot, “Automatic verification of relational properties on programs manipulating algebraic dataypes”, since September 2021, Thomas Jensen and Thomas Genet.
- PhD: Louis Noizet, “Necro : la sémantique sans y laisser les os”, October 2019–September 2022, Alan Schmitt.
- PhD: Guillaume Ambal, “Skeletal Semantics Transformations”, September 2019–October 2022, Sergueï Lenglet and Alan Schmitt.

- PhD: Adam Khayam, “A Meta-Approach to Describe Effectful and Distributed Semantics”, September 2019–October 2022, Tamara Rezk and Alan Schmitt.
- PhD in progress: Vincent Rébiscoul, “Analyses Statiques pour Sémantiques Squelettiques”, since September 2020, Thomas Jensen and Alan Schmitt.
- L3 internship: Charles de Haro, “A fixpoint solver for dependent functions in OCaml”, April 2022–May 2022, Benoît Montagu.
- PhD in progress: Gautier Raimondi, “Secure Compilation against Side-channel Attacks”, since September 2020, Thomas Jensen and Frédéric Besson.
- PhD in progress: Jean-Loup Hatchikian-Houdot, “Security-enhancing compiler against side-channel attacks“, since Octobre 2021, Guillaume Hiet and Frédéric Besson.

10.2.2 Juries

PhD and HDR defenses

- Alan Schmitt, jury member (reviewer) for the PhD defense of Gaby Sampaio, March 2022, Imperial College
- Alan Schmitt, jury member (reviewer) for the PhD defense of Olivier Nicole, April 2022, École Normale Supérieure – PSL
- Alan Schmitt, jury member (reviewer) for the PhD defense of Dara Ly, December 2022, Université d’Orléans
- Alan Schmitt, jury member for the PhD defense of Aurèle Barrière, December 2022, Université de Rennes
- Sandrine Blazy, jury member (president) for the PhD defense of Daniel De Almeida Braga, Rennes 1 University. 14/12/2022.
- Sandrine Blazy, jury member (president) for the PhD defense of Adam Khayam, Rennes 1 University, 30/11/2022.
- Sandrine Blazy, jury member (president) for the PhD defense of Jean-Joseph Marty, Rennes 1 University, 17/11/2022.
- Sandrine Blazy, jury member for the PhD defense of Basile Clément, ENS - University Paris SL, 09/09/2022.
- Thomas Genet, jury member (reviewer) for the PhD defense of Pierre Lermusiaux, Lorraine University, 8/09/2022.

Hiring and Promotion Committees

- Sandrine Blazy: president of hiring committee for an Associate Professor in Software Security, Univ Rennes 1 (ISTIC) / IRISA, Spring 2022.
- Sandrine Blazy: president of hiring committee for a Professor, Univ Rennes 1 (ESIR) / IRISA, Spring 2022.
- Delphine Demange: member of the selection committee for Inria Rennes Moyens Incitatifs 2022, Inria RBA, April 2022.
- Delphine Demange: member of hiring committee for an Associate Professor in Software Security, Univ Rennes 1 (ISTIC) / IRISA, Spring 2022.
- Delphine Demange: member of hiring committee for an Associate Professor in Formal Methods and Security, ENS Paris-Saclay / LMF, Spring 2022.
- Thomas Jensen: member of the CRCN/ISFP jury for junior scientists at Inria Rennes.

Other

- Name, role, date and period

10.3 Popularization

10.3.1 Internal or external Inria responsibilities

- Alan Schmitt in Interim Director of the OCaml Software Foundation since October 2022

10.3.2 Education

- Licence : Sandrine Blazy, Programmation de Confiance, 57h, L3, Université Rennes 1, France.
- Master : Sandrine Blazy, Mechanized semantics, 35h, M1, Université Rennes 1, France.
- Licence : Delphine Demange, Spécialité Informatique 1 - Algorithmique et Complexité Expérimentale, 40h, L1, Université Rennes 1, France.
- Licence : Delphine Demange, Informatique 1 - Programmation Impérative en Java, 40h, L1, Université Rennes 1, France.
- Licence : Delphine Demange, Complément en Informatique 1 - Programmation Impérative en Java, 12h, L1, Université Rennes 1, France.
- Licence : Delphine Demange, Mise à Niveau en Informatique - Programmation en Java, 12h, L3, Université Rennes 1, France.
- Licence : Thomas Genet, Spécialité Informatique 1 - Algorithmique et Complexité Expérimentale, 39h, L1, Université Rennes 1, France.
- Licence : Thomas Genet, Génie logiciel, 54h, L2, Université Rennes 1, France.
- Master : Thomas Genet, Analyse et Conception Formelles, 38h, M1, Université Rennes 1, France
- Master : Thomas Genet, Blockchain, 12h, M2, Université Rennes 1, France
- Licence : Benoît Montagu, Programmation de Confiance, 36h, L3, Université Rennes 1, France.
- Master : Benoît Montagu, Analyse et Conception Formelles, 24h, M1, Université Rennes 1, France
- Master : Alan Schmitt, Preparation of Agregation exam, 80h, M2, ENS Rennes, France.
- Master : Benoît Montagu, Paradigmes de programmation (Preparation for Agregation exam), 12h, M2, ENS Rennes, France.
- Licence : Frédéric Besson, Programmation fonctionnelle, 28h, L3, Insa Rennes, France.
- Master : Simon Castellan, Sémantique et analyse statique, 6h, M2, ENS Rennes, France.
- Master : Thomas Jensen,Sémantique et analyse statique, 14h, M2, ENS Rennes, France.

11 Scientific production

11.1 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.

11.2 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). DOI: 10.1145/3571202. URL: https://hal.inria.fr/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). URL: https://hal.archives-ouvertes.fr/hal-03286443.

International peer-reviewed conferences

[16] G. Ambal, S. Lenglet and A. Schmitt. 'Certified Abstract Machines for Skeletal Semantics'. In: CPP 2022 - 11th ACM SIGPLAN International Conference on Certified Programs and Proofs. Philadelphia, United States, 17th Jan. 2022, pp. 1–13. DOI: 10.1145/3497775.3503676. URL: https://hal.inria.fr/hal-03466807.

[17] G. Ambal, S. Lenglet, A. Schmitt and C. Noûs. 'Certified Derivation of Small-Step From Big-Step Skeletal Semantics'. In: PPDP 2022 - 24th International Symposium on Principles and Practice of Declarative Programming. Tbilisi, Georgia, 20th Sept. 2022, pp. 1–48. DOI: 10.1145/3551357.3551384. URL: https://hal.inria.fr/hal-03768820.

[18] S. Bautista, T. Jensen and B. Montagu. 'Lifting Numeric Relational Domains to Algebraic Data Types'. In: *Static Analysis - 29th International Symposium, SAS 2022 Auckland, New Zealand, December 5–7, 2022, Proceedings*. SAS 2022 - 29th International Symposium on Static Analysis. Vol. LNCS-13790. Lecture Notes in Computer Science. Auckland, New Zealand: Springer Nature Switzerland, 2nd Dec. 2022, pp. 104–134. DOI: 10.1007/978-3-031-22308-2_6. URL: https://hal.inria.fr/hal-03906375.

[19] M. Biernacka, D. Biernacki, S. Lenglet and A. Schmitt. 'Non-Deterministic Abstract Machines'. In: CONCUR 2022 - 33rd International Conference on Concurrency Theory. Varsovie, Poland, 13th Sept. 2022, pp. 1–24. DOI: 10.4230/LIPIcs.CONCUR.2022.7. URL: https://hal.inria.fr/hal-03772712.

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

[21] A. Khayam, L. Noizet and A. Schmitt. 'A Faithful Description of ECMAScript Algorithms'. In: PPDP 2022 - 24th International Symposium on Principles and Practice of Declarative Programming. Tbilisi, Georgia: ACM, 20th Sept. 2022, pp. 1–14. DOI: 10.1145/3551357.3551381. URL: https://hal.inria.fr/hal-03782992.

[22] L. Noizet and A. Schmitt. 'Semantics in Skel and Necro'. In: ICTCS 2022 - Italian Conference on Theoretical Computer Science. CEUR Workshop Proceedings. Rome, Italy, 9th Sept. 2022, pp. 1–17. URL: https://hal.inria.fr/hal-03784478.

[23] 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, 7th Aug. 2022, pp. 1–23. URL: https://hal.inria.fr/hal-03888082.

[24] K. Zandberg, E. Baccelli, S. Yuan, F. Besson and J.-P. Talpin. 'Femto-Containers: Lightweight Virtualization and Fault Isolation For Small Software Functions on Low-Power IoT Microcontrollers'. In: Middleware 2022 - 23rd ACM/IFIP International Conference Middleware. quebec, Canada, 7th Nov. 2022, pp. 1–12. DOI: 10.1145/3528535.3565242. URL: https://hal.inria.fr/hal-03888109.

Conferences without proceedings

[25] J.-L. Hatchikian-Houdot. 'Constant Time Secure Embedded Systems Through Hardware/Software Cooperation'. In: RESSI 2022 - Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information. Chambon-sur-Lac, France, 10th May 2022, pp. 1–3. URL: https://hal.archives-ouvertes.fr/hal-03882701.

Reports & preprints

[26] 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, 28th Nov. 2022. URL: https://hal.inria.fr/hal-03765357.

[27] T. Jensen. *Updated SPARTA SRIA (Roadmap v3): Roadmap for the SPARTA Cybersecurity Competence Network*. INRIA, 23rd Sept. 2022. URL: https://hal.inria.fr/hal-03907545.

11.3 Cited publications

[28] O. F. Andreescu, T. Jensen, S. Lescuyer and B. Montagu. 'Inferring Frame Conditions with Static Correlation Analysis'. In: *Proc. ACM Program. Lang.* 3.POPL (Jan. 2019). DOI: 10.1145/3290360. URL: https://doi.org/10.1145/3290360.

[29] T. Haudebourg, T. Genet and T. P. Jensen. 'Regular language type inference with term rewriting'. In: *Proc. ACM Program. Lang.* 4.ICFP (2020), 112:1–112:29. DOI: 10.1145/3408994.

[30] G. Klein, M. Norrish, T. Sewell, H. Tuch, S. Winwood, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt and R. Kolanski. 'seL4: Formal Verification of an OS Kernel'. In: *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles - SOSP '09*. ACM Press, 2009. DOI: 10.1145/1629575.1629596.

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

[32] B. Montagu and T. P. Jensen. 'Stable Relations and Abstract Interpretation of Higher-order Programs'. In: *Proc. ACM Program. Lang.* 4.ICFP (2020), 119:1–119:30. DOI: 10.1145/3409001.

[33] B. Montagu and T. P. Jensen. 'Trace-based control-flow analysis'. In: *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 20211*. Ed. by S. N. Freund and E. Yahav. ACM, 2021, pp. 482–496. DOI: 10.1145/3453483.3454057.