RESEARCH CENTRE

Inria Center
at Université Côte d'Azur

2022
ACTIVITY REPORT

Team
INDES

Secure Diffuse Programming

Inria teams are typically groups of researchers working on the definition of a common project, and objectives, with the goal to arrive at the creation of a project-team. Such project-teams may include other partners (universities or research institutions)

DOMAIN

Networks, Systems and Services, Distributed Computing

THEME

Distributed programming and Software engineering

Inria

Contents

Team INDES

Creation of the Team: 2022 June 22

Keywords

Computer sciences and digital sciences

A1.3. – Distributed Systems

A2. – Software

A2.1. – Programming Languages

A2.1.1. – Semantics of programming languages

A2.1.3. – Object-oriented programming

A2.1.4. – Functional programming

A2.1.7. – Distributed programming

A2.1.9. – Synchronous languages

A2.1.12. – Dynamic languages

A2.2.1. – Static analysis

A2.2.5. – Run-time systems

A2.2.9. – Security by compilation

A4.3.3. – Cryptographic protocols

A4.6. – Authentication

A4.7. – Access control

Other research topics and application domains

B6.3.1. – Web

B6.4. – Internet of things

B9.5.1. – Computer science

B9.10. – Privacy

1 Team members, visitors, external collaborators

Research Scientists

- Manuel Serrano [Team leader, INRIA, Senior Researcher, HDR]
- Ilaria Castellani [INRIA, Researcher]
- Guillaume Combette [CEA, Researcher, from Jul 2022]
- Lautaro Lecumberry [AIRBUS CYBERSECURITY GMBH, Researcher, from Oct 2022]
- Tamara Rezk [INRIA, Senior Researcher, HDR]

Faculty Members

- Gérard Berry [COLLEGE DE FRANCE, HDR]
- Marc Feeley [UNIV MONTREAL]
- Robert Findler [NORTHWESTERN UNIVERSITY, from Dec 2022]
- David Naumann [STEVENS INSTITUTE OF TECHNOLOGY, from Oct 2022]
- Andreas Sabelfeld [UNIV TECH CHALMERS, from Sep 2022]

PhD Students

- Davide Davoli [UNIV COTE D'AZUR, from Oct 2022]
- Mohamad El Laz [POLE EMPLOI, from Feb 2022]
- Mohamad El Laz [INRIA, until Jan 2022]
- Jayanth Krishnamurthy [INRIA]

Administrative Assistant

- Nathalie Bellesso [INRIA]

External Collaborator

- Bertrand Petit [Pole Emploi]

2 Overall objectives

The goal of the Indes team is to study models for diffuse computing and develop languages for secure diffuse applications. Diffuse applications, of which Web 2.0 applications are a notable example, are the new applications emerging from the convergence of broad network accessibility, rich personal digital environment, and vast sources of information. Strong security guarantees are required for these applications, which intrinsically rely on sharing private information over networks of mutually distrustful nodes connected by unreliable media.

Diffuse computing requires an original combination of nearly all previous computing paradigms, ranging from classical sequential computing to parallel and concurrent computing in both their synchronous / reactive and asynchronous variants. It also benefits from the recent advances in mobile computing, since devices involved in diffuse applications are often mobile or portable.

The Indes team contributes to the whole chain of research on models and languages for diffuse computing, going from the study of foundational models and formal semantics to the design and implementation of new languages to be put to work on concrete applications. Emphasis is placed on

correct-by-construction mechanisms to guarantee correct, efficient and secure implementation of high-level programs. The research is partly inspired by and built around Hop, the web programming model proposed by the former Mimosa team, which takes the web as its execution platform and targets interactive and multimedia applications.

3 Research program

3.1 Parallelism, concurrency, and distribution

Concurrency management is at the heart of diffuse programming. Since the execution platforms are highly heterogeneous, many different concurrency principles and models may be involved. Asynchronous concurrency is the basis of shared-memory process handling within multiprocessor or multicore computers, of direct or fifo-based message passing in distributed networks, and of fifo- or interrupt-based event handling in web-based human-machine interaction or sensor handling. Synchronous or quasi-synchronous concurrency is the basis of signal processing, of real-time control, and of safety-critical information acquisition and display. Interfacing existing devices based on these different concurrency principles within Hop or other diffuse programming languages will require better understanding of the underlying concurrency models and of the way they can nicely cooperate, a currently ill-resolved problem.

3.2 Web, functional, and reactive programming

We are studying new paradigms for programming Web applications that rely on multi-tier functional programming. We have created a Web programming environment named Hop. It relies on a single formalism for programming the server-side and the client-side of the applications as well as for configuring the execution engine.

Hop is a functional language based on the SCHEME programming language. That is, it is a strict functional language, fully polymorphic, supporting side effects, and dynamically type-checked. Hop is implemented as an extension of the BIGLOO Scheme compiler that we develop. In the past, we have extensively studied static analyses (type systems and inference, abstract interpretations, as well as classical compiler optimizations) to improve the efficiency of compilation in both space and time.

As a Hop DSL, we have created HipHop, a synchronous orchestration language for web and IoT applications. HipHop facilitates the design and programming of complex web/IoT applications by smoothly integrating three computation models and programming styles that have been historically developed in different communities and for different purposes: *i) Transformational programs* that simply compute output values from input values, with comparatively simple interaction with their environment; *ii)* asynchronous concurrent programs that perform interactions between their components or with their environment with uncontrollable timing, using typically network-based communication; and *iii)* synchronous reactive programs that react to external events in a conceptually instantaneous and deterministic way.

3.3 Security of diffuse programs

The main goal of our security research is to provide scalable and rigorous language-based techniques that can be integrated into multi-tier compilers to enforce the security of diffuse programs. Research on language-based security has been carried on before in former Inria teams. In particular previous research has focused on controlling information flow to ensure confidentiality.

Typical language-based solutions to these problems are founded on static analysis, logics, provable cryptography, and compilers that generate correct code by construction. Relying on the multi-tier programming language Hop that tames the complexity of writing and analysing secure diffuse applications, we are studying language-based solutions to prominent web security problems such as code injection and cross-site scripting, to name a few.

4 Application domains

4.1 Web

The Web is the natural application domain of the team. We are designing and implementing multitier languages for helping the development of Web applications. We are creating static and dynamic analyses for Web security. We are conducting empirical studies about privacy preservation on the Web.

4.2 Internet of Things

More recently, we have started focusing on *Internet of Things* (IoT) applications. They share many similarities with Web applications so most of the methodologies and expertises we have developed for the Web apply to IoT but the restricted hardware resources made available by many IoT devices demand new developments and new research explorations.

5 Highlights of the year

This section should rather be called “Lowlights”, given its negative assessments.

We point out several issues and institutional dysfunctions which impaired and slowed down our team activity, and also badly affected the general atmosphere in the research centre and in the institute at large, causing a great deal of anxiety and strain in the scientific, technical and administrative staff.

- The deployment of the Eksae information system was highly problematic, substantially hindering the activity of our administrative staff, forcing them to tedious duplications and rendering some of their tasks almost impossible. This also had repercussions on researchers, depriving them of a long-term vision of their budget and generating more constraints and delays in purchases and mission reimbursements, for both team members and their visitors. Even recurrent events, such as the “Journées scientifiques” and the hiring/promotion campaigns, suffered from delays and bad advertisement due to the excessive workload of the technical and administrative staff.
- The presentation of the institute given in our general direction (DG) addresses and publications, such as the “Rapport d’activité 2021” (Question d’avenir), offered a distorted and unbalanced image of our institute, hiding the primary role of research and emphasising only the most “trendy” and “applicable” research activities, while dismissing many others for which our institute gained its prestigious reputation.
- The shift of our institute towards a “service agency”, as prefigured by the inclusion of all INRIA centres within universities and the increasing pressure on researchers to participate in teaching within these universities, makes the future of INRIA as a national research institute and the status of its scientific and administrative staff extremely uncertain.
- A deep distress within the staff, as well as an increasing distrust towards the DG, has arisen by effect of the dismissive attitude of the DG towards the consultative bodies of the institute, and particularly towards the Evaluation Commission (CE), whose role is essential for the quality of our hiring and promotion processes as well as for our prospective scientific reflexions.

6 New software and platforms

6.1 New software

6.1.1 Bigloo

Keyword: Compilers

Functional Description: Bigloo is a Scheme implementation devoted to one goal: enabling Scheme based programming style where C(++) is usually required. Bigloo attempts to make Scheme

practical by offering features usually presented by traditional programming languages but not offered by Scheme and functional programming. Bigloo compiles Scheme modules. It delivers small and fast stand alone binary executables. Bigloo enables full connections between Scheme and C programs and between Scheme and Java programs.

Release Contributions: modification of the object system (language design and implementation), new APIs (alsa, flac, mpg123, avahi, csv parsing), new library functions (UDP support), new regular expressions support, new garbage collector (Boehm's collection 7.3alpha1).

URL: http://www-sop.inria.fr/teams/indes/fp/Bigloo/

Contact: Manuel Serrano

Participant: Manuel Serrano

6.1.2 Hop

Keywords: Programming language, Multimedia, Iot, Web 2.0, Functional programming

Scientific Description: The Hop programming environment consists in a web broker that intuitively combines in a single architecture a web server and a web proxy. The broker embeds a Hop interpreter for executing server-side code and a Hop client-side compiler for generating the code that will get executed by the client.

An important effort is devoted to providing Hop with a realistic and efficient implementation. The Hop implementation is validated against web applications that are used on a daily-basis. In particular, we have developed Hop applications for authoring and projecting slides, editing calendars, reading RSS streams, or managing blogs.

Functional Description: Multitier web programming language and runtime environment.

URL: http://hop.inria.fr

Contact: Manuel Serrano

Participant: Manuel Serrano

6.1.3 IFJS

Name: Infomation Flow monitor inlining for JavaScript

Keyword: Cybersecurity

Functional Description: The IFJS compiler is applied to JavaScript code. The compiler generates JavaScript code instrumented with checks to secure code. The compiler takes into account special features of JavaScript such as implicit type coercions and programs that actively try to bypass the inlined enforcement mechanisms. The compiler guarantees that third-party programs cannot (1) access the compiler internal state by randomizing the names of the resources through which it is accessed and (2) change the behaviour of native functions that are used by the enforcement mechanisms inlined in the compiled code.

URL: http://www-sop.inria.fr/indes/ifJS/

Contact: Tamara Rezk

6.1.4 Hiphop.js

Name: Hiphop.js

Keywords: Web 2.0, Synchronous Language, Programming language

Functional Description: HipHop.js is an Hop.js DLS for orchestrating web applications. HipHop.js helps programming and maintaining Web applications where the orchestration of asynchronous tasks is complex.

URL: http://hop-dev.inria.fr/hiphop

Contact: Manuel Serrano

6.1.5 Server-Side Protection against Third Party Web Tracking

Keywords: Privacy, Web Application, Web, Architecture, Security by design, Program rewriting techniques

Functional Description: We present a new web application architecture that allows web developers to gain control over certain types of third party content. In the traditional web application architecture, a web application developer has no control over third party content. This allows the exchange of tracking information between the browser and the third party content provider.

To prevent this, our solution is based on the automatic rewriting of the web application in such a way that the third party requests are redirected to a trusted third party server, called the Middle Party Server. It may be either controlled by a trusted party, or by a main site owner and automatically eliminates third-party tracking cookies and other technologies that may be exchanged by the browser and third party server

URL: http://www-sop.inria.fr/members/Doliere.Some/essos/

Contact: Francis Doliére Some

6.1.6 webstats

Name: Webstats

Keywords: Web Usage Mining, Statistic analysis, Security

Functional Description: The goal of this tool is to perform a large-scale monthly crawl of the top Alexa sites, collecting both inline scripts (written by web developers) and remote scripts, and establishing the popularity of remote scripts (such as Google Analytics and jQuery). With this data, we establish whether the collected scripts are actually written in a subset of JavaScript by analyzing the different constructs used in those scripts. Finally, we collect and analyze the HTTP headers of the different sites visited, and provide statistics about the usage of HTTPOnly and Secure cookies, and the Content Security Policy in top sites.

URL: https://webstats.inria.fr

Contact: Francis Doliére Some

6.1.7 Skini Node.js (ISS)

Name: Platform for creation and execution for audience participative music

Keywords: Music, Interaction, Web Application, Synchronous Language

Functional Description: Skini is a platform for designing and performing collaborative music. It is based on two musical concepts: pattern and orchestration. The orchestration is designed using HipHop.js.

Release Contributions: Can be use for performance and création.

Author: Bertrand Petit

Contact: Bertrand Petit

7 New results

We have pursued the development of Hop and our study on efficient JavaScript implementations as well as our development of analyses for distributed language sessions and security.

7.1 Design and Implementation of Dynamic Languages

7.1.1 JavaScript Sealed Classes

Participants: Manuel Serrano.

We proposed the JavaScript *Sealed Classes*, which differ from regular classes in a few ways that allow ahead-of-time (AoT) compilers to implement them more efficiently. Sealed classes are compatible with the rest of the language so that they can be combined with all other structures, including regular classes, and can be gradually integrated into existing code bases.

Sealed classes trade a little bit of the dynamicity of JavaScript classes for faster and more predictable execution. All the benchmarks we tested benefit from sealed classes. Some benefit from a code size reduction and others benefit from speedup. Some benefit from both.

Sealed classes are compatible with the rest of the JavaScript runtime system. They can be passed to functions, returned by them, stored in data structures, and they can be used as the super classes of sealed and ordinary classes. Thus, in existing programs, sealed classes can gradually replace those classes that naturally respect the restrictions they impose. Infringements to the rules of sealed classes are detected, so that sealing classes does not present the risk of silently corrupting operational programs.

The dynamic semantics of sealed classes that do not raise errors is identical to that of regular classes. They can therefore already be used by unmodified JavaScript engines, although in this case there is no runtime acceleration. To benefit from this acceleration, we have modified the AoT Hopc compiler. We have shown that the average speedup due to sealed classes is of 19% on a variety of programs using classes. We have detailed this implementation in a conference paper [19]. It is simple and required less than 1,000 new lines of code for the compiler and a few hundred lines of code for the runtime system. Sealed classes deliver better performance than regular classes *and* they are easy to implement.

7.1.2 Semi-Automatic Verification of TypeScript Type Declarations

Participants: Robby Findler, Manuel Serrano.

The DefinitelyTyped repository hosts type declarations for thousands of JavaScript libraries. Given the lack of formal connection between the types and the corresponding code, a natural question is *are the types right?* An equally important question, as DefinitelyTyped and the libraries it supports change over time, is *how can we keep the types from becoming wrong?*

To tackle this problem, we have created Scotty, a tool that detects mismatches between the types and code in the DefinitelyTyped repository. More specifically, Scotty checks each package by converting its types into contracts and installing the contracts on the boundary between the library and its test suite. Running the test suite in this environment can reveal mismatches between the types and the JavaScript code. As automation and generality are both essential if such a tool is going to remain useful in the long term, we focus on techniques that sacrifice completeness, instead preferring to avoid false positives. Scotty currently handles about 26% of the 8806 packages on DefinitelyTyped (61% of the packages whose code is available and whose test suite passes).

Perhaps unsurprisingly, running the tests with these contracts in place revealed many errors in DefinitelyTyped. More surprisingly, despite the inherent limitations of the techniques we use, this exercise led to one hundred accepted pull requests that fix errors in DefinitelyTyped, demonstrating the value of this approach for the long-term maintenance of DefinitelyTyped. It also revealed a number of lessons about working in the JavaScript ecosystem and how details beyond the semantics of the language can be surprisingly important. Best of all, it also revealed a few places where programmers preferred incorrect types, suggesting some avenues of research to improve TypeScript.

Scotty, its design, its architecture, and also its limits, have been described in a publication [14].

7.2 Session Types

Participants: Ilaria Castellani.

Session types describe communication protocols involving two or more participants by specifying the sequence of exchanged messages and their functionality (sender, receiver and type of carried data). They may be viewed as the analogue, for concurrency and distribution, of data types for sequential computation. Originally conceived as a static analysis technique for a variant of the π-calculus, session types have been progressively embedded into a range of functional, concurrent, and object-oriented programming languages.

The aim of session types is to ensure safety properties for sessions, such as the *absence of communication errors* (no type mismatch in exchanged data) and *deadlock-freedom* (no standstill until all participants are terminated). When describing multiparty protocols, session types often target also the liveness property of *progress* or *lock-freedom* (no participant waits forever).

While binary sessions can be described by a single session type, multiparty sessions require two kinds of types: a *global type* that describes the whole session protocol, and *local types* that describe the individual contributions of the participants to the protocol. The key requirement to achieve safety properties such as deadlock-freedom is that the local types of the processes implementing the participants be obtained as projections from the same global type. To ensure progress, global types must satisfy additional well-formedness requirements.

What makes session types particularly attractive is that they offer several advantages at once: 1) static safety guarantees, 2) automatic check of protocol implementation correctness, based on local types, and 3) a strong connection with linear logics and with concurrency models such as communicating automata, graphical choreographies and message-sequence charts.

During the past year we have further investigated the relationship between multiparty session types and concurrency models, focussing on Event Structures [27], a canonical model for concurrent computation with explicit notions of causality and concurrency. We have also addressed the issue of input races in multiparty sessions, and proposed a new type system that accepts some kinds of "innocuous" input races, thus enlarging the class of protocols that can be specified by session types.

Like most of our previous work on this subject, this research has been pursued in collaboration with colleagues from the Universities of Eastern Piedmont and Turin.

7.2.1 Event Structure Semantics for Synchronous Multiparty Sessions

We proposed a denotational semantics for multiparty session calculi by means of Event Structures (ESs), a well-known concurrency model introduced in the early 80's [28, 26].

We considered a core multiparty session calculus with *synchronous communication*, where sessions are described as networks of sequential processes (each process implementing a participant), equipped with standard global types. We proposed an interpretation of networks as *Flow Event Structures* (FESs) [25], a subclass of Winskel's Stable Event Structures [28], as well as an interpretation of global types as *Prime Event Structures* (PESs) [26], the simplest class of ESs. Concurrency between network communications may be directly reflected in the events of the associated FES. On the other hand, since global types are sequential specifications, which are not able to explicitly represent concurrency between communications, the events of the associated PES need to be defined as equivalence classes of communication

sequences up to *permutation equivalence*. We showed that when a network is typable with a global type, the FES semantics of the former is equivalent to the PES semantics of its type.

This work has been published in the journal JLAMP [12].

7.2.2 Asynchronous Sessions with Input Races

The original papers on multiparty session types imposed strong restrictions on the syntax of global types, requiring all initial communications in the branches of a choice to have the same sender and the same receiver, and every other participant to be independent from the choice, i.e., to have the same behaviour in all branches. Although these were useful simplifying assumptions in order to achieve multiparty session correctness, they limited the expressiveness of global types, ruling out relevant protocols. For this reason, more permissive choice constructors were investigated in subsequent work. However *input races*, namely the possibility for a receiver to choose between inputs from different senders, continued to be viewed as problematic and to be forbidden by typing. As a consequence, common protocols such as a server shared by different clients could not be specified by global types.

In the paper [16] we propose a more flexible type system for asynchronous multiparty sessions, which allows two kinds of innocuous input races, which we call respectively *confluent races* and *fake races*, while still rejecting dangerous races that could lead to deadlock or starvation.

7.3 Security

7.3.1 Security Analyses for XSS

Participants: Héloise Maurel, Tamara Rezk.

Cross-site Scripting (XSS) is one of the most dangerous software weaknesses due to its constant popularity through the years. Several dynamic and static approaches for detection and prevention have been explored in the past. In this work, we explore static approaches to detect XSS vulnerabilities using neural networks. We compare two different code representations based on Natural Language Processing (NLP) and Programming Language Processing (PLP) and experiment with models based on different neural network architectures for static analysis detection in PHP and Node.js. We train and evaluate the models using synthetic databases. Using the generated PHP and Node.js databases, we compare our results with a well-known static analyzer for PHP code, ProgPilot, and a known scanner for Node.js, AppScan static mode. Our analyzers using neural networks improve on the results of existing tools in all cases.

This work was part of the PhD thesis of Héloise Maurel, defended in November 2022. The work is described in her PhD thesis [23] and in two publications [18] and a journal article to appear.

7.3.2 Binary Analysis for Secret Erasure

Participants: Tamara Rezk.

We tackle the problem of designing efficient binary-level verification for a subset of information flow properties encompassing constant-time and secret-erasure. These properties are crucial for cryptographic implementations, but are generally not preserved by compilers. Our proposal builds on relational symbolic execution enhanced with new optimizations dedicated to information flow and binary-level analysis, yielding a dramatic improvement over prior work based on symbolic execution. We implement a prototype, Binsec/Rel, for bug-finding and bounded-verification of constant-time and secret-erasure, and perform extensive experiments on a set of 338 cryptographic implementations, demonstrating the benefits of our approach. Using Binsec/Rel, we also automate two prior manual studies on preservation of constant-time and secret-erasure by compilers for a total of 4148 and 1156 binaries respectively. Interestingly, our analysis highlights incorrect usages of volatile data pointers for secret erasure and

shows that scrubbing mechanisms based on volatile function pointers can introduce additional register spilling which might break secret-erasure. We also discovered that gcc -O0 and backend passes of clang introduce violations of constant-time in implementations that were previously deemed secure by a state-of-the-art constant-time verification tool operating at LLVM level, showing the importance of reasoning at binary-level. We have published this work in an important journal for computer security, TOPS [13].

8 Partnerships and cooperations

8.1 International initiatives

8.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program

HipHopSec

Title: Secure Reactive IoT Programming

Duration: 2020 ->

Coordinator: Robby Findler (`robby@eecs.northwestern.edu`)

Partners: Northwestern University (Chicago) (USA)

Inria contact: Manuel Serrano

Summary: Nowadays most applications are distributed, that is, they run on several computers: a mobile device for the graphical user interface a gateway for storing data in a local area; a remote server of a large cloud platform for resource demanding computing; an object connected to Internet in the IoT (Internet of Things); etc. For many different reasons, this makes programming much more difficult than it was when only a single computer was involved:

- Applications are composed of extensive lists of diverse components, each coming with their own specification and imposing their own constraints on application development.
- Due to the distributed nature of the applications, developers have to implement appropriate communication protocols, which is difficult to do correctly and securely.
- Communicating applications need to resort to parallelism to handle requests from their clients with acceptable latency. No matter whether it is multi-threading (as in Java) or asynchronous programming (as in JavaScript/Node.js), this style of programming is notoriously difficult and error-prone.

The Indes, Northwestern, and Collège de France teams are studying programming languages and have each created complementary solutions that address the aforementioned problems. Combined together, they could lead to a robust and secure execution environment for the web and IoT programming. Indes will bring its expertise in secure web programming, Collège de France its expertise in synchronous reactive programming, Northwestern its expertise in secure execution environments and run-time validation of security properties of program executions. Finally Northwestern will contribute with its expertise in medical descriptions, which will be the main application domain of the secure execution environment the participants aim to develop.

The main objective of the collaboration is the development of a robust and secure integrated programming environment for reactive applications suitable for web and IoT applications. The programming of medical prescriptions will be our favored application domain. We will base our work on three pillars: Hop.js, the contract system designed for the Racket language, and HipHop.js, a domain specific language for reactive programming within Hop.js.

- HipHop.js has currently minimal integration with Hop.js and a rudimentary programming environment. We will continue the development of HipHop.js with the goal of turning it into a a usable and reliable platform.

- The formal semantics of HipHop.js is based on rewriting logics, automata theory and Boolean equations. Thus, HipHop.js programs can be verified using existing techniques based on the satisfiability of logic formulas. Such techniques have been widely used for synchronous reactive programs, but never before in the more dynamic world of web or medical applications.
- Supporting medical prescriptions as programs requires not only a language with special syntactic abstractions to match the notations of the medical domain, but also a fundamentally new way to think about prescription vs. computer programs. For example, medical personnel often modifies prescriptions in the middle of a treatment. In linguistic terms this requires that the programming language in use supports the ability to pause a program while it is running, modify its code, and restart it from the point of the pause but with the modified version of the code, this in a guaranteed consistent way. We hope to build such a programming language, with a semantics inspired by synchronous-reactive programming in the style of HipHop.js but tailored to the medical domain.
- Contracts state precise properties of the interfaces of components and validate them at run time. Over the last fifteen years, Racket developers, including those dealing with the language itself, have used contracts extensively to validate properties that range from simple type-like constraints to partial functional correctness and even security. Our goal is to design and implement a contract system for Hop/HipHop.js that is as expressive as that of Racket. Hop/HipHop.js is based on Javascript, a different linguistic setting than that of Racket; however, existing work on Javascript proxies and macros has resulted in encouraging preliminary results on contracts for higher-order functions and objects in Javascript. We aim at lifting and extending these results to Hop/HipHop.js. Given an expressive contract system for Hop/HipHop.js, we will investigate: (i) how to state and enforce security policies for Hop/HipHop.js applications with contracts; and (ii) how different compilation and implementation techniques can alleviate existing performance issues of applications, a current weakness that impedes the widespread adoption of contracts.
- Improving the quality of the code requires support for testing. S. You (working with C. Dimoulas and R. Findler) is working on improving automated testing techniques. So far he has discovered a new theoretical result showing how to use concolic testing for higher-order functions. This result may have applications for testing in JavaScript and we are hopeful that we can leverage it to Hop.js.

8.2 European initiatives

8.2.1 H2020 projects

SPARTA SPARTA project on cordis.europa.eu

Title: Strategic programs for advanced research and technology in Europe

Coordianator: CEA, FRANCE

Duration: From February 1, 2019 to June 30, 2022

Partners:

- INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (INRIA), France
- CESNET ZAJMOVE SDRUZENI PRAVNICKYCH OSOB (CESNET), Czechia
- JOANNEUM RESEARCH FORSCHUNGSGESELLSCHAFT MBH (JOANNEUM RESEARCH), Austria
- NAUKOWA I AKADEMICKA SIEC KOMPUTEROWA - PANSTWOWY INSTYTUT BADAWCZY (NASK), Poland
- TARTU ULIKOOL (UNIVERSITY OF TARTU), Estonia
- MYKOLO ROMERIO UNIVERSITETAS (MYKOLAROMERIS UNIVERSITY), Lithuania

- LATVIJAS MOBILAIS TELEFONS SIA, Latvia
- SECURITY MADE IN LETZEBUERG (SMILE), Luxembourg
- FRAUNHOFER GESELLSCHAFT ZUR FORDERUNG DER ANGEWANDTEN FORSCHUNG EV (FHG), Germany
- FUNDACION TECNALIA RESEARCH & INNOVATION (TECNALIA), Spain
- TECHNISCHE UNIVERSITAET MUENCHEN (TUM), Germany
- THALES SIX GTS FRANCE SAS (THALES SIX GTS France), France
- COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES (CEA), France
- STOWARZYSZENIE POLSKA PLATFORMA BEZPIECZENSTWA WEWNETRZNEGO (PPBW), Poland
- INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON (INSA LYON), France
- SAP SE, Germany
- FORTISS GMBH, Germany
- LUXEMBOURG INSTITUTE OF SCIENCE AND TECHNOLOGY (LIST), Luxembourg
- VYSOKE UCENI TECHNICKE V BRNE (BRNO UNIVERSITY OF TECHNOLOGY), Czechia
- FUNDACION CENTRO DE TECNOLOGIAS DE INTERACCION VISUAL Y COMUNICACIONES VICOMTECH (VICOM), Spain
- INDRA SISTEMAS SA (INDRA), Spain
- INSTITUT MINES-TELECOM, France
- RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITAT BONN, Germany
- UNIVERSITE DU LUXEMBOURG (uni.lu), Luxembourg
- CONSIGLIO NAZIONALE DELLE RICERCHE (CNR), Italy
- "NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS" ("NCSR "D"), Greece
- LIETUVOS KIBERNETINIU NUSIKALTIMU KOMPETENCIJU IR TYRIMU CENTRAS (LITHUANIAN CYBERCRIME CENTER OF EXCELLENCE FOR TRAINING RESEARCH & EDUCATIO), Lithuania
- KENTRO MELETON ASFALEIAS (CENTER FORSECURITY STUDIES CENTRE D'ETUDES DE SECURITE), Greece
- INDRA FACTORIA TECNOLOGICA SL, Spain
- UNIVERSITAT KONSTANZ (UKON), Germany
- LEONARDO - SOCIETA PER AZIONI (LEONARDO), Italy
- KAUNO TECHNOLOGIJOS UNIVERSITETAS (UNIVERSITY OF TECHNOLOGY, KAUNAS), Lithuania
- TECHNIKON FORSCHUNGS- UND PLANUNGSGESELLSCHAFT MBH (TECHNIKON), Austria
- ITTI SP ZOO (ITTI), Poland
- DIREZIONE GENERALE PER LE TECNOLOGIE DELLE COMUNICAZIONI E LA SICUREZZA INFORMATICA - ISTITUTO SUPERIORE DELLE COMUNICAZIONI E DELLE TECNOLOGIE DELL'INFORMAZIONE (DG TCSI-ISCOM), Italy
- GENEROLO JONO ZEMAICIO LIETUVOS KARO AKADEMIJA (GENERAL JONAS ZEMAITISMILITARY ACADEMY OF LITHUANIA), Lithuania
- FUNDACIO EURECAT (EURECAT), Spain
- CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI (CNIT), Italy
- CENTRALESUPELEC, France

- YES WE HACK (YWH), France
- INSTITUTO SUPERIOR TECNICO (IST), Portugal
- SECRETARIAT GENERAL DE LA DEFENSE ET DE LA SECURITE NATIONALE (SGDSN), France
- UNIVERSITE DE NAMUR ASBL (UNamur), Belgium
- INOV INSTITUTO DE ENGENHARIA DE SISTEMAS E COMPUTADORES INOVACAO (INOV), Portugal
- CENTRE D'EXCELLENCE EN TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION (CETIC), Belgium
- CZ.NIC, ZSPO (CZ.NIC), Czechia
- CONSORZIO INTERUNIVERSITARIO NAZIONALE PER L'INFORMATICA (CINI), Italy

Inria contact: Thomas Jensen

Coordinator:

Summary: In the domain of Cybersecurity Research and innovation, European scientists hold pioneering positions in fields such as cryptography, formal methods, or secure components. Yet this excellence on focused domains does not translate into larger-scale, system-level advantages. Too often, scattered and small teams fall short of critical mass capabilities, despite demonstrating world-class talent and results. Europe's strength is in its diversity, but that strength is only materialised if we cooperate, combine, and develop common lines of research. Given today's societal challenges, this has become more than an advantage' an urgent necessity. Various approaches are being developed to enhance collaboration at many levels. Europe's framework programs have sprung projects in cybersecurity over the past thirty years, encouraging international cooperation and funding support actions. More recently, the Cybersecurity PPP has brought together public institutions and industrial actors around common roadmaps and projects. While encouraging, these efforts have highlighted the need to break the mould, to step up investments and intensify coordination. The SPARTA proposal brings together a unique set of actors at the intersection of scientific excellence, technological innovation, and societal sciences in cybersecurity. Strongly guided by concrete and risky challenges, it will setup unique collaboration means, leading the way in building transformative capabilities and forming world-leading expertise centres. Through innovative governance, ambitious demonstration cases, and active community engagement, SPARTA aims at re-thinking the way cybersecurity research is performed in Europe across domains and expertise, from foundations to applications, in academia and industry.

8.2.2 ANR CISC

Participants: Ilaria Castellani, Tamara Rezk, Manuel Serrano.

The CISC project (Certified IoT Secure Compilation) is funded by the ANR for 42 months, ending in September 2023. The goal of the CISC project is to provide strong security and privacy guarantees for IoT applications by means of a language to orchestrate IoT applicatoins from the microcontroller to the cloud. Tamara Rezk coordinates this project, and Manuel Serrano, Ilaria Castellani and Nataliia Bielova participate in the project. The partners of this project are Inria teams Celtique, Indes and Privatics, and Collège de France.

9 Dissemination

Participants: Ilaria Castellani, Tamara Rezk, Manuel Serrano.

9.1 Promoting scientific activities

9.1.1 Scientific events: organisation

General chair, scientific chair Tamara Rezk organized and chaired PLMW at PLDI'22.

9.1.2 Scientific events: selection

Member of the conference program committees

- Ilaria Castellani participated in the program committees of:
  - PLACES'22: 13th Workshop on Programming Language Approaches to Concurrency- and Communication-cEntric Software
  - CONCUR'22: 33rd International Conference on Concurrency Theory
- Tamara Rezk participated in the program committees of:
  - IEEE S&P'22: IEEE Security and Privacy Symposium
  - ACM CCS'22: ACM Communications on Computer Security
  - ACSAC'22: Annual Computer Security Applications Conference
  - TheWebConf'22: The Web Conference
  - ICDCS'22: IEEE International Conference on Distributed Computing Systems
- Manuel Serrano participated in the program committees of:
  - ECOOP'22: European Conference on Object-Oriented Programming
  - ICFP'22: ACM International Conference on Functional Programming
  - PROGRAMMING'22: Programming Conference

9.1.3 Journal

Member of the editorial boards

- Ilaria Castellani was guest editor for a special issue of the journal JLAMP [11].
- Manuel Serrano is a member of the Steering Committee for the conference and journal "Programming".

9.1.4 Invited talks

- Ilaria Castellani gave the invited talk *Global types and event structure semantics for asynchronous multiparty sessions* at the workshop ICE'22.
- Tamara Rezk was the keynote speaker for the European Symposium on Security and Privacy 2022. Her talk was entitled: *2022: Have Transient Execution Attacks Been Fully Solved?*.
- Manuel Serrano gave the following keynotes and invited talks:
  - *Static compilation of JavaScript*, MPLR'22 (Brussels, Belgium)
  - *Of JavaScript Ahead-Of-Time Compilation Performance*, StrangeLoop'22 (St Louis, USA)
  - *Of JavaScript Ahead-Of-Time Compilation Performance*, MEMOCODE'22

9.1.5 Research administration

- Tamara Rezk is part of the *bureau du CP* at INRIA Sophia Antipolis.
- Manuel Serrano is vice-head of the Inria Evaluation Committee. As such he co-organizes all the grants, promotion juries and the juries of the national recruiting campaigns. He also co-organizes all the team evaluation seminars.

9.2 Teaching - Supervision - Juries

9.2.1 Teaching

Tamara Rezk taught 56 hours ETD of courses in Université Côte d'Azur, master level.

9.2.2 Supervision

- PhD in progress: Jayanth Krishnamurthy, Secure Reactive Web Programming, 12/09/2018, Manuel Serrano.
- PhD in progress : Ignacio Tiraboschi, Security analyses, 1/9/2020, Tamara Rezk and Xavier Rival.
- PhD in progress: Guillaume Combette, Binary Analyses, 1/6/2022, Sébastien Bardin and Tamara Rezk.
- PhD in progress: Davide Davoli, Secure randomization, 1/10/2022, Martin Avanzini and Tamara Rezk.
- PhD defended: Mohamad El Laz, Provable encryption schemes for distributed systems [21], Benjamin Grégoire and Tamara Rezk.
- PhD defended: Héloise Maurel, Deep Learning applied on Web Security [23], Tamara Rezk.
- PhD defended: Adam Khayam, A Meta-Approach to Describe Effectful and Distributed Semantics [22], Tamara Rezk and Alan Schmitt.

9.2.3 Juries

- Ilaria Castellani was the chair of the jury of the CONCUR 2022 Test-of-Time Award [17].
- Ilaria Castellani participated as a reviewer in the jury of the PhD thesis of Elli Anastasiadi (supervisors: Luca Aceto and Anna Ingolfsdottir), Reykjavik University, October 2022.
- Tamara Rezk participated in the following juries:
  - Phd Jury (Reviewer): Natalia Kulatova (supervisor: Karthikeyan Bhargavan), ENS Ulm 2022
  - CSD Jury: Jonathan Brossard (supervisors: Nadia Lammari, Véronique Legrand), CNAM, 2022
  - CSD Jury: Swarn Priva (supervisors: Yves Bertot, Benjamin Grégoire), Université Côte d'Azur, 2022
  - External Reviewer for the European Research Council, ERC Advanced Grant 2021-Call, 2022
  - CRCN for Handicaped People, Jury member (Sophia Antipolis competition), Inria 2022
  - CRCN Jury member (Sophia Antipolis competition), Inria 2022
- Manuel Serrano was an examiner of the PhD thesis of Aurèle Barrière.

9.3 Popularization

9.3.1 Interventions

Tamara Rezk participated at the W@PLDI panel at PLDI'22.

10 Scientific production

10.1 Major publications

[1] N. Bielova and T. Rezk. 'A Taxonomy of Information Flow Monitors'. In: *International Conference on Principles of Security and Trust (POST 2016)*. Ed. by F. Piessens and L. Viganò. Vol. 9635. LNCS - Lecture Notes in Computer Science. Eindhoven, Netherlands: Springer, Apr. 2016, pp. 46–67. DOI: 10.1007/978-3-662-49635-0_3. URL: https://hal.inria.fr/hal-01348188.

[2] G. Boudol and I. Castellani. 'Noninterference for Concurrent Programs and Thread Systems'. In: *Theoretical Computer Science* 281.1 (2002), pp. 109–130.

[3] G. Boudol, Z. Luo, T. Rezk and M. Serrano. 'Reasoning about Web Applications: An Operational Semantics for HOP'. In: ACM TRANSACTIONS ON PROGRAMMING LANGUAGES AND SYSTEMS (TOPLAS) 34.2 (2012).

[4] S. Capecchi, I. Castellani and M. Dezani-Ciancaglini. 'Information Flow Safety in Multiparty Sessions'. In: *Mathematical Structures in Computer Science*. Special Issue: EXPRESS'11 26.8 (2015), p. 43. DOI: 10.1017/S0960129514000619. URL: https://hal.inria.fr/hal-01237236.

[5] I. Castellani, M. Dezani-Ciancaglini and P. Giannini. 'Concurrent Reversible Sessions'. In: *CONCUR 2017 - 28th International Conference on Concurrency Theory*. Vol. 85. CONCUR 2017. Roland Meyer and Uwe Nestmann. Berlin, Germany, Sept. 2017, pp. 1–17. DOI: 10.4230/LIPIcs.CONCUR.2017.30. URL: https://hal.inria.fr/hal-01639845.

[6] C. Fournet and T. Rezk. 'Cryptographically sound implementations for typed information-flow security'. In: *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*. 2008, pp. 323–335.

[7] M. Ngo, F. Piessens and T. Rezk. 'Impossibility of Precise and Sound Termination-Sensitive Security Enforcements'. In: *SP 2018 - IEEE Symposium on Security and Privacy*. San Francisco, United States: IEEE, May 2018, pp. 496–513. DOI: 10.1109/SP.2018.00048. URL: https://hal.inria.fr/hal-01928669.

[8] M. Serrano and G. Berry. 'Multitier Programming in Hop - A first step toward programming 21st-century applications'. In: *Communications of the ACM* 55.8 (Aug. 2012), pp. 53–59. DOI: 10.1145/2240236.2240253. URL: http://cacm.acm.org/magazines/2012/8/153796-multitier-programming-in-hop/abstract.

[9] M. Serrano and V. Prunet. 'A Glimpse of Hopjs'. In: *21th* ACM SIGPLAN INT'L CONFERENCE ON FUNCTIONAL PROGRAMMING (ICFP). Nara, Japan, Sept. 2016, pp. 188–200. URL: http://dx.doi.org/10.1145/2951913.2951916.

[10] D. F. Somé, N. Bielova and T. Rezk. 'On the Content Security Policy Violations due to the Same-Origin Policy'. In: *26th International World Wide Web Conference, 2017 (WWW 2017)* (Apr. 2017). DOI: 10.1145/3038912.3052634. URL: https://hal.inria.fr/hal-01649526.

10.2 Publications of the year

International journals

[11] I. Castellani, P. D'Argenio, M. R. Mousavi and A. Sokolova. 'Preface to the special issue on Open Problems in Concurrency Theory'. In: *Journal of Logical and Algebraic Methods in Programming* 130 (Jan. 2023), p. 100823. DOI: 10.1016/j.jlamp.2022.100823. URL: https://hal.inria.fr/hal-03970947.

[12] I. Castellani, M. Dezani-Ciancaglini and P. Giannini. 'Event structure semantics for multiparty sessions'. In: *Journal of Logical and Algebraic Methods in Programming* 131 (Feb. 2023). DOI: 10.1016/j.jlamp.2022.100844. URL: https://hal.inria.fr/hal-03940191.

[13] L.-A. Daniel, S. Bardin and T. Rezk. 'Binsec/Rel: Symbolic Binary Analyzer for Security with Applications to Constant-Time and Secret-Erasure'. In: *ACM Transactions on Privacy and Security* (2022). URL: https://hal.inria.fr/hal-03833598.

[14] J. Hoeflich, R. B. Findler and M. Serrano. 'Highly illogical, Kirk: spotting type mismatches in the large despite broken contracts, unsound types, and too many linters'. In: *Proceedings of the ACM on Programming Languages* 6.OOPSLA2 (31st Oct. 2022), pp. 479–504. DOI: 10.1145/3563305. URL: https://hal.inria.fr/hal-03920363.

[15] H. Maurel, S. Vidal and T. Rezk. 'Statically identifying XSS using deep learning'. In: *Science of Computer Programming* 219 (July 2022), p. 102810. DOI: 10.1016/j.scico.2022.102810. URL: https://hal.inria.fr/hal-03684437.

International peer-reviewed conferences

[16] I. Castellani, M. Dezani-Ciancaglini and P. Giannini. 'Asynchronous Sessions with Input Races'. In: *EPTCS, Proceedings of the 13th International Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software*. PLACES 2022. Vol. 356. Munich (Allemagne), Germany, 24th Mar. 2022, pp. 12–23. DOI: 10.4204/EPTCS.356.2. URL: https://hal.inria.fr/hal-03940160.

[17] I. Castellani, P. Gastin, O. Kupferman, M. Randour and D. Sangiorgi. 'CONCUR Test-Of-Time Award 2022'. In: 33rd International Conference on Concurrency Theory (CONCUR 2022). Vol. 243. Warsaw (Poland), Poland, 6th Sept. 2022, 1:1–1:3. DOI: 10.4230/LIPIcs.CONCUR.2022.1. URL: https://hal.inria.fr/hal-03970965.

[18] H. Maurel, S. Vidal and T. Rezk. 'Comparing the Detection of XSS Vulnerabilities in Node.js and a Multi-tier JavaScript-based Language via Deep Learning'. In: ICISSP 2022 - 8th International Conference on Information Systems Security and Privacy. Virtual, France, 9th Feb. 2022. DOI: 10.5220/0010980800003120. URL: https://hal.inria.fr/hal-03576267.

[19] M. Serrano. 'JavaScript Sealed Classes'. In: 36th European Conference on Object-Oriented Programming (ECOOP 2022). Berlin, Germany, 6th June 2022. URL: https://hal.inria.fr/hal-03920356.

[20] I. Tiraboschi, T. Rezk and X. Rival. 'Sound Symbolic Execution via Abstract Interpretation and its Application to Security'. In: *Lecture Notes in Computer Science*. 24th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2023). Vol. 13881. Verification, Model Checking, and Abstract Interpretation 24th International Conference, VMCAI 2023, Boston, MA, USA, January 16–17, 2023, Proceeding. Boston, MA, United States: Springer Nature Switzerland, 17th Jan. 2023, pp. 267–295. DOI: 10.1007/978-3-031-24950-1_13. URL: https://hal.science/hal-03942146.

Doctoral dissertations and habilitation theses

[21] M. El Laz. 'Provable encryption schemes for distributed systems'. Université Côte d'Azur, 30th Mar. 2022. URL: https://theses.hal.science/tel-03814201.

[22] A. Khayam. 'A Meta-Approach to Describe Effectful and Distributed Semantics'. Université Rennes 1, 30th Nov. 2022. URL: https://theses.hal.science/tel-03969183.

[23] H. Maurel. 'Deep learning applied on Web security: Statically Identifying Web vulnerabilities using Deep Learning'. INRIA : Institut national de recherche en sciences et technologies du numérique; Université Cote d'Azur, 14th Nov. 2022. URL: https://hal.inria.fr/tel-03849284.

Reports & preprints

[24] M. El Laz, A. Hevia and T. Rezk. *Tracking Information Flow by Mapping Broadcast Encryption Subgroups to Security Lattices*. Inria, 19th Jan. 2022. URL: https://hal.inria.fr/hal-03537962.

10.3 Cited publications

[25] G. Boudol and I. Castellani. 'Permutation of transitions: an event structure semantics for CCS and SCCS'. In: *REX: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*. Ed. by J. W. de Bakker, W. P. de Roever and G. Rozenberg. Vol. 354. LNCS. Springer, 1988, pp. 411–427.

[26] M. Nielsen, G. Plotkin and G. Winskel. 'Petri Nets, Event Structures and Domains, Part I'. In: *Theoretical Computer Science* 13.1 (1981), pp. 85–108.

[27] G. Winskel. 'An introduction to event structures'. In: *REX: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*. Ed. by J. W. de Bakker, W. P. de Roever and G. Rozenberg. Vol. 354. LNCS. Heidelberg: Springer, 1988, pp. 364–397.

[28] G. Winskel. 'Events in Computation'. PhD thesis. University of Edinburgh, 1980.