RESEARCH CENTRE

Sophia Antipolis - Méditerranée

2020 ACTIVITY REPORT

Project-Team INDES

Secure Diffuse Programming

DOMAIN

Networks, Systems and Services, Distributed Computing

THEME

Distributed programming and Software engineering

Contents

Project-Team INDES

Creation of the Team: 2009 January 01, updated into Project-Team: 2010 July 01

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]
- Tamara Rezk [Inria, Researcher, HDR]

Faculty Member

- Gérard Berry [Collège de France, Emeritus, HDR]

PhD Students

- Lesly Ann Daniel [CEA]
- Mohamad El Laz [Inria, from Dec 2020]
- Jayanth Krishnamurthy [Inria]
- Heloise Maurel [Inria]
- Bertrand Petit [Pôle Emploi, until Aug 2020]

Technical Staff

- Yoon Seok Ko [Inria, Engineer]

Administrative Assistant

- Nathalie Bellesso [Inria]

External Collaborator

- Marc Feeley [Université de Montréal - Canada]

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 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

5.1 Awards

Lesly Ann Daniel received the award “Jeunes Talents France” L’Oréal - UNESCO pour les femmes et la science" for her work on automated program analysis for security.

6 New software and platforms

6.1 Binsec/REL

Name: Automatic Symbolic Analysis of Constant Time at Binary Level

Keyword: Cybersecurity

Functional Description: Binsec/Rel is an extension of the binary analysis plateform Binsec that implements relational symbolic execution (RelSE) for constant-time (CT) verification.

URL: https://github.com/binsec/Rel

Contact: Tamara Rezk

6.2 New software

6.2.1 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, 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

6.2.2 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

6.2.3 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

6.2.4 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

Contacts: Manuel Serrano, Gérard Berry

6.2.5 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

6.2.6 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

Contacts: Francis Doliére Some, Tamara Rezk, Nataliia Bielova

6.2.7 Skini

Name: Platform for creation and execution for audience participative music

Keywords: Music, Interaction, Web Application, Synchronous Language

Functional Description: Skini is a platform form designing et performing collaborative music. It is based on two musical concept: pattern and orchestration. The orchestration is design using HipHop.js.

Release Contributions: Can be use for performance and création.

Contact: Bertrand Petit

7 New results

We have pursued the development of Hop and our study on efficient and secure JavaScript implementations.

7.1 JavaScript Implementation

Inline caches and hidden classes are two essential components for closing the performance gap between static languages such as Java, Scheme, or ML and dynamic languages such as JavaScript or Python. They rely on the observation that for a particular object access located at a particular point of the program, the shapes, usually referred to as *the hidden classes*, of accessed objects are likely to be the same. Taking benefit of that invariant, they replace the expensive lookup the semantics of these languages normally demand with one test, the *inline cache*, and a memory read indexed by an offset computed during the last cache miss. These optimizations are essential but they are not general enough to cope with JavaScript's proxies. In particular, when the property name is itself unknown statically, inline cache-based optimizations always take a slow path.

We have conducted an analysis that shown how to generalize inline caches to cope with an unknown property name. We have exposed the general principle of the extension that we have then implemented. We have conducted experiments using a modified version of the Hop JavaScript compiler that demonstrated how the optimization is crucial for improving the performance of proxy objects (as they naturally use dynamic property names extensively). This evaluation report shown that the modified Hop outperforms all other implementations of the language, including the most efficient commercial ones, by a factor ranging from 2× to 100×. Even better, our optimizations are applicable to existing compilers as they require only straightforward changes to runtime data structures; no complex analyses are required.

This study has been conducted in collaboration with Robby Findler in the context of the HipHopSec Inria associated team. It has been published in a scientific conference [16].

We have also worked on new optimizations to apply to the Hop ahead-of-time compiler. Its new version is due to be released during the first trimester.

7.2 Web Reactive Programming

7.2.1 HipHop.js

HipHop is a reactive JavaScript DSL on Hop. In the domain of reactive programming, causality errors - wherein a signal's presence/absence may depend circularly on another signal giving rise to causality issues. Before the discussed work started on the HipHop compiler, the support by the HipHop run-time for causality errors was limited to pointing out the presence of causal dependencies as an error during a reaction by the HipHop machine and rejection of the program. Our work started on providing a more elaborate debugging support for HipHop programmers.

HipHop and HipHop applications have been published in a conference paper this year [14].

This year, we have completed the implementation of the HipHop.js web reactive synchronous language and we have mainly focused our effort on the programming and debugging environment.

Once the run-time rejects a program due to causality error, the HipHop program which is compiled into set of nets called net-list, will have the nets that are not participating in a reaction. We take these nets as a digraph and proceed with Depth First search algorithm to identify strongly connected components among them using Tarjan's algorithm. The nets in strongly connected components are the ones participating in Causal cycles. Further, since a strongly connected component can have smaller sub components which are strongly connected, we used the technique proposed by F. Bourdoncle to further refine the strongly connected components to smaller sub components pointing to smallest causality cycle that can be easily managed by the programmer. We take these smallers subcomponents and process the information available in the nets to identify and point out the source locations that are contributing to causality errors. The programmers can resolve these smaller cycles and then incrementally identify next bigger cycle if any. Once the cycle locations are identified, we presented their locations with visual markers on Emacs editor. Further, this approach was extended to popular IDEs like Vscode and Atom editors. We are preparing a research paper that will describe this work.

Recently visual programming tool kits like Blockly have become very popular with non - cs background programmers. We are extending HipHop programming to Blockly so that musicians can use HipHop for interactive music generation. We intend to bring in the causality debugging support to Blockly programmers as a future work.

7.2.2 Interactive Music Composition

Skini is a programming methodology and an execution environment for interactive structured music. With this system, the composer *programs* his scores in the HipHop synchronous reactive language. They are then executed, or *played*, in live concerts, in interaction with the audience. The system aims at helping composers to find a good balance between the determinism of the compositions and the nondeterminism of the interactions with the public. Each execution of a Skini score yields to a different but aesthetically consistent interpretation.

This work raises many questions in the musical fields. *How to combine composition and interaction? How to control the musical style when the audience influences what is to play next? What are the possible connections with generative music?* These are important questions for the Skini system but they are out of the scope of this work that focuses exclusively on the computer science aspects of the system. From that perspective, the main questions are *how to program the scores* and *in which language?* General purpose languages are inappropriate because their elementary constructs (*i.e.,* variables, functions, loops, etc.) do not match the constructions needed to express music and musical constraints. We show that synchronous programming languages are a much better fit because they rely on temporal constructs that can be directly used to represent musical scores and because their malleability enables composers to experiment easily with artistic variations of their initial scores.

We have published a journal paper [12] that focuses on scores programming. It exposes the process a composer should follow from his very first musical intuitions up to the generation of a musical artifact. The paper presents some excerpts of the programming of a classical music composition that it then precisely relates to an actual recording. Examples of techno music and jazz are also presented, with audio artifact, to demonstrate the versatility of the system. Finally, brief presentations of past live concerts are presented as an evidence of viability of the system. A second paper [17] focuses on the production of generative music with Skini.

7.3 Session Types

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 an enhanced version of the π-calculus, session types have been subsequently 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). Multiparty session types often target also the liveness property of *progress* or *lock-freedom* (no participant waits forever), which is stronger than deadlock-freedom.

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 the absence of communication errors and 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.

We have pursued our work on multiparty session types along the two directions described below, in collaboration with colleagues from the Universities of Turin and Eastern Piedmont.

7.3.1 Multiparty Sessions with Internal Delegation

We have investigated a new form of *delegation* for multiparty session calculi. Usually, delegation allows a session participant to appoint a participant in another session to act on her behalf. In this view, delegation is an inter-session mechanism which requires session interleaving. Hence it falls outside the descriptive power of global types, which specify single multiparty sessions. As a consequence, properties such as deadlock-freedom or lock-freedom are difficult to ensure in the presence of delegation. In our work, we adopt a different view of delegation, by allowing participants to delegate tasks to each other within the same multiparty session. This way, delegation occurs within a single session (whence the name “internal

delegation") and may be captured by its global type. We present a session type system based on global types with internal delegation, and show that it ensures the usual safety properties of multiparty sessions, together with a progress property.

This work has been published in a special issue of the journal *Theoretical Computer Science* dedicated to Maurice Nivat [11].

7.3.2 Global Types and Event Structure Semantics for Asynchronous Multiparty Sessions

In previous work we explored the relationship between synchronous multiparty sessions and Event Structures (ESs), a well-known concurrency model introduced in the early 80's. We considered a core multiparty session calculus 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), a subclass of Winskel's Stable Event Structures, as well as an interpretation of global types as *Prime Event Structures* (PESs), the simplest class of ESs. Since the syntax of global types does not allow all the concurrency among communications to be explicitly represented, 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 by a global type, the FES semantics of the former is equivalent, in a precise technical sense, to the PES semantics of the latter.

In the work [21], we undertake a similar endeavour in the asynchronous setting. This involves devising a new notion of global type for asynchronous sessions. The type system for asynchronous sessions is expected to be more permissive than the one for synchronous sessions. For instance, consider a session with two participants each of which wishes to first send a message to the other one and then receive a message from the other one. This session is stuck if communication is synchronous but not if communication is asynchronous.

We start by considering a core session calculus as in the synchronous case, where networks are endowed with a queue and they act on this queue by performing outputs or inputs: an output stores a message in the queue, while an input fetches a message from the queue. The intuition for our new *asynchronous global types* is quite simple: to split communications in the type into outputs and inputs, and to equip the type with a queue, thus mimicking very closely the behaviour of asynchronous networks. The well-formedness conditions for global types must now take into account also the content of the queue. Essentially, this amounts to requiring that each input appearing in the type be justified by a preceding output in the type or by a message in the queue, and vice versa, that each output in the type or message in the queue be matched by a corresponding input in the type.

The contribution of [21] is twofold: 1) We propose an original type system for asynchronous multiparty sessions, which accounts for asynchronous communication in a more natural way than existing approaches, while remaining decidable. Our type system ensures the classical safety properties of sessions as well as progress; 2) We present an Event Structure semantics for asynchronous sessions and for asynchronous global types, and we show that these two semantics agree.

This paper has been submitted for journal publication.

7.4 Security Analysis of ElGamal Implementations

The ElGamal encryption scheme is not only the most extensively used alternative to RSA, but is also almost exclusively used in voting systems as an effective homomorphic encryption scheme. Being easily adaptable to a wide range of cryptographic groups, the ElGamal encryption scheme enjoys homomorphic properties while remaining semantically secure. This is subject to the upholding of the Decisional Diffie-Hellman (DDH) assumption on the chosen group. We analyze 26 libraries that implement the ElGamal encryption scheme and discover that 20 of them are semantically insecure as they do not respect the Decisional Diffie-Hellman (DDH) assumption. From the five libraries that do satisfy the DDH assumption, we identify and compare four different message encoding and decoding techniques.

7.5 Timing Leaks

Timing leaks have been a major concern for the security community. A common approach is to prevent secrets from affecting the execution time, thus achieving security with respect to a strong, local attacker

who can measure the timing of program runs. However, this approach becomes restrictive as soon as programs branch on a secret, it becomes ineffective both when compiler passes re-introduces branches on secrets, or when an attacker exploits speculations. We have studied timing leaks in all of these different scenarios.

7.5.1 ClockworK: Timing Leaks in popular IoT application platforms

This work focuses on timing leaks under remote execution. A key difference is that the remote attacker does not have a reference point of when a program run has started or finished, which significantly restricts attacker capabilities. We propose an extensional security characterization that captures the essence of remote timing attacks. We identify patterns of combining clock access, secret branching, and output in a way that leads to timing leaks. Based on these patterns, we design Clockwork, a monitor that rules out remote timing leaks. We implement the approach for JavaScript, leveraging JSFlow, a state-of-the-art information flow tracker. We demonstrate the feasibility of the approach on case studies with IFTTT, a popular IoT app platform, and VJSC, an advanced JavaScript library for e-voting.

7.5.2 Spectre attacks

The constant-time programming discipline (CT) is a software-based countermeasure used for protecting high assurance cryptographic implementations against timing side-channel attacks. Constant time is effective (it protects against many known attacks), rigorous (it can be formalized using program semantics), and amenable to automated verification. Yet, the advent of microarchitectural attacks makes constant-time as it exists today far less useful. This work lays foundations for constant-time programming in the presence of speculative and out-of-order execution. We present an operational semantics and a formal definition of constant-time programs in this extended setting. Our semantics eschews formalization of microarchitectural features (that are instead assumed under adversary control), and yields a notion of constant-time that retains the elegance and tractability of the usual notion. We demonstrate the relevance of our semantics in two ways: First, by contrasting existing Spectre-like attacks with our definition of constant-time. Second, by implementing a static analysis tool, Pitchfork, which detects violations of our extended constant-time property in real world cryptographic libraries

7.5.3 Binsec/Rel

Writing CT code is challenging as it demands to reason about pairs of execution traces (2- hypersafety property) and it is generally not preserved by the compiler, requiring binary-level analysis. Unfortunately, current verification tools for CT either reason at higher level (C or LLVM), or sacrifice bug-finding or bounded-verification, or do not scale. We tackle the problem of designing an efficient binary-level verification tool for CT providing both bug-finding and bounded-verification. The technique 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, and perform extensive experiments on a set of 338 cryptographic implementations, demonstrating the benefits of our approach in both bug-finding and bounded-verification. Using BINSEC/REL, we also automate a previous manual study of CT preservation by compilers. Interestingly, we discovered that gcc -O0 and backend passes of clang introduce violations of CT in implementations that were previously deemed secure by a state-of-the-art CT verification tool operating at LLVM level, showing the importance of reasoning at binary-level.

8 Bilateral contracts and grants with industry

8.1 Bilateral grants with industry

The ANSWER project (Advanced aNd Secured Web Experience and seaRch) is lead by the QWANT search engine and the Inria Sophia Antipolis Méditerranée research center. This proposal is the winner of the "Grand Challenges du Numérique" (BPI) and aims to develop the new version of the search engine http://www.qwant.com with radical innovations in terms of search criteria, indexed content and

privacy of users. Nataliia Bielova, Manuel Serrano and Tamara Rezk are involved in this project. The project started on January 1, 2018. In the context of this project, we got

- with Arnaud Legout from the DIANA project-team a funding for a 3 years Ph.D. student to work on Web tracking technologies and privacy protection. Imane Fouad, a former Indes member, was hired to work on this project.
- a funding for 18 months Postdoc to work on Web application security. Yoon Seok Ko has worked on this project as a postdoc.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Inria associate team not involved in an IIL

HipHopSec

Title: *Secure Reactive IoT Programming*

Duration: 2020 - 2021

Coordinator: Manuel Serrano

Partners: Northwestern University (Chicago, United States).

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 its 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 College 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, College de France its expertise in synchronous reactive programming, Northwestern its expertise 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 result 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 from 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.

9.1.2 Inria international partners

We are collaborating with Universidad de Chile (Santiago Chile) and Universidad Nacional del Centro de la Provincia de Buenos Aires (Argentina) on the Detection strategies based on Software Metrics for Multitier JavaScript. Our main collaborators are Alexandre Bergel and Santiago Vidal. This project was due to end in 2019 but because of the political situation in Chile first, and Covid second, we have had to suspend our collaboration for two years. We hope we will be able this year to complete the initiated studies. If the context permit, M. Serrano will visit A. Bergel in December 2021. In the meantime S. Vidal collaborates with T. Rezk and H. Maurel.

Declared Inria international partners Marc Feeley from University of Montréal is a recurrent visitor of the team. Unfortunately, his visit of the year has been canceled for obvious reasons but it did not end our collaboration. We are participating in weekly meetings with him and his student.

9.2 European initiatives

9.2.1 FP7 & H2020 Projects

SPARTA

Title: Special projects for advanced research and technology in Europe

Duration: *February 2019 - January 2022*

Coordinator: *CEA*

Partners:

- *C*ENTRE D'EXCELLENCE EN TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION (Belgium)
- *C*ESNET ZAJMOVE SDRUZENI PRAVNICKYCH OSOB (Czech Republic)
- *C*OMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES (France)
- *C*ONSIGLIO NAZIONALE DELLE RICERCHE (Italy)
- *C*ONSORZIO INTERUNIVERSITARIO NAZIONALE PER L'INFORMATICA (Italy)
- *C*ONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI (Italy)
- *C*Z.NIC, ZSPO (Czech Republic)
- *D*IREZIONE GENERALE PER LE TECNOLOGIE DELLE COMUNICAZIONI E LA SICUREZZA INFORMATICA - ISTITUTO SUPERIORE DELLE COMUNICAZIONI E DELLE TECNOLOGIE DELL'INFORMAZIONE (Italy)
- *F*RAUNHOFER GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V. (Germany)
- *F*UNDACIO EURECAT (Spain)
- *F*UNDACION CENTRO DE TECNOLOGIAS DE INTERACCION VISUAL Y COMUNICACIONES VICOMTECH (Spain)
- *F*UNDACION TECNALIA RESEARCH & INNOVATION (Spain)
- *G*ENEROLO JONO ZEMAICIO LIETUVOS KARO AKADEMIJA (Lithuania)
- *I*NDRA SISTEMAS SA (Spain)
- *I*NOV INESC INOVACAO - INSTITUTO DE NOVAS TECNOLOGIAS (Portugal)
- *I*NSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON (France)
- *I*NSTITUTO SUPERIOR TECNICO (Portugal)
- *I*TTI SP ZOO (Poland)
- *J*OANNEUM RESEARCH FORSCHUNGSGESELLSCHAFT MBH (Austria)
- *K*AUNO TECHNOLOGIJOS UNIVERSITETAS (Lithuania)
- *K*ENTRO MELETON ASFALEIAS (Greece)
- *L*EONARDO - SOCIETA PER AZIONI (Italy)
- *L*IETUVOS KIBERNETINIU NUSIKALTIMU KOMPETENCIJU IR TYRIMU CENTRAS (Lithuania)
- *L*UXEMBOURG INSTITUTE OF SCIENCE AND TECHNOLOGY (Luxembourg)
- *M*YKOLO ROMERIO UNIVERSITETAS (Lithuania)
- *N*ATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS" (Greece)
- *N*AUKOWA I AKADEMICKA SIEC KOMPUTEROWA - PANSTWOWY INSTYTUT BADAWCZY (Poland)
- *S*ECRETARIAT GENERAL DE LA DEFENSE ET DE LA SECURITE NATIONALE (France)
- *S*TOWARZYSZENIE POLSKA PLATFORMA BEZPIECZENSTWA WEWNETRZNEGO (Poland)
- *T*ARTU ULIKOOL (Estonia)
- *T*ECHNIKON FORSCHUNGS- UND PLANUNGSGESELLSCHAFT MBH (Austria)
- *T*ECHNISCHE UNIVERSITAET MUENCHEN (Germany)

- *T*HALES SIX GTS FRANCE SAS (France)
- *U*NIVERSITAT KONSTANZ (Germany)
- *U*NIVERSITE DE NAMUR ASBL (Belgium)
- *U*NIVERSITE DU LUXEMBOURG (Luxembourg)
- *V*YSOKE UCENI TECHNICKE V BRNE (Czech Republic)

Inria contact: *Thomas Jensen*

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.</abstract>*

9.3 National initiatives

9.3.1 ANR CISC

The CISC project (Certified IoT Secure Compilation) is funded by the ANR for 42 months, starting in April 2018. 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.

9.3.2 ANR PrivaWeb

The PrivaWeb project (Privacy Protection and ePrivacy Compliance for Web Users) is funded by the ANR JCJC program for 48 months, started in December 2018. PrivaWeb aims at developing new methods for detection of new Web tracking technologies and new tools to integrate in existing Web applications that seamlessly protect privacy of users.

Nataliia Bielova (PRIVATICS project-team) coordinates this project.

9.3.3 PIA ANSWER

The ANSWER project (Advanced aNd Secured Web Experience and seaRch) is funded by PIA program for 36 months, starting January 1, 2018. The aim of the ANSWER project is to develop the new version of the http://www.qwant.com search engine by introducing radical innovations in terms of search criteria as well as indexed content and users' privacy. The partners of this project include QWANT and Inria teams Wimmics, Indes, Neo and Diana.

10 Dissemination

10.1 Promoting Scientific Activities

10.1.1 Scientific Events: Organisation

- Tamara Rezk organized the 2020 Shonan meeting 159 on Web Security, together with Limin Jia (Carnegie Melon University) and Sukyoung Ryu (KAIST), inviting experts in web security from academy and industry. The meeting was canceled a week before its date due to COVID and moved to March 2022;

General Chair, Scientific Chair

- Ilaria Castellani was the co-chair (together with Mohammad Reza Mousavi) of the workshop TRENDS 2020, the annual event of the IFIP WG1.8 on Concurrency Theory, which took place virtually on September 5, 2020, in association with the CONCUR 2020 conference (these events were due to take place in Vienna but were turned into virtual events because of Covid). https://concurrency-theory.org/events/workshops/trends

Member of the Conference Program Committees

- Manuel Serrano was participating in the Program Committee of the ECOOP'20 conference.
- Ilaria Castellani served in the Program Committee of the workshop PLACES 2020. http://places20.by.di.fc.ul.pt/
- Tamara Rezk served in the Program Committee of ASIACCS 2020, MADWeb 2020, SecDev 2020, ACISP 2020, and SecWeb 2020.

10.1.2 Invited Talks

- Manuel Serrano gave a keynote conference at the FDL conference about Web Reactive Programming and HipHop.js (http://www.fdl-conference.org/).

10.1.3 Leadership within the Scientific Community

- Ilaria Castellani was the chair of the IFIP TC1 WG 1.8 on Concurrency Theory from June 2014 to December 2020. http://www.ifip-tc1.org/ https://concurrency-theory.org/organizations/ifip
- Tamara Rezk is a member of the Steering Committee of the PriSC workshop.

10.1.4 Research Administration

- Manuel Serrano is vice-head of the Inria Evaluation Committee. As such he co-organizes all the grants, promotions juries and the juries of the national recruiting campaigns. He also co-organizes all the team evaluation seminars.
- Ilaria Castellani was the chair of the "jury d'admissibilité" for the recruitment of junior researchers in the INRIA Centre of Grenoble Rhône Alpes.
- Ilaria Castellani is a member of INRIA's "Comité Parité et Égalité des Chances". In the Centre of INRIA Sophia Antipolis, she is a member of the "Comité Scientifique du Colloquium". Within UCA (Université Côte d'Azur), she is a member of the organising committee of the "Forum Numerica" seminar series.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Tamara Rezk taught two courses (master level) at University of Nice-Sophia Antipolis: Web Security (28 ETD) and Cryptographic proofs (28 ETD).

10.2.2 Supervision

- Postdoc: Yoon Seok Ko, Secure JavaScript, 1/10/2018-, Tamara Rezk, Manuel Serrano.
- PhD in progress: Jayanth Krishnamurthy, Secure Reactive Web Programming, 12/09/2018, Manuel Serrano.
- PhD in progress: Bertrand Petit, Musique Massivement Interactive, 12/09/2017, Manuel Serrano. PhD defended in July 2020.
- PhD in progress : Héloïse Maurel, Statically Identifying Security Vulnerabilities using Deep Learning 1/10/2018, Tamara Rezk
- PhD in progress : Mohamad Ellaz, Implementation and analysis of cryptographic libraries, 1/12/2017, Benjamin Grégoire and Tamara Rezk
- PhD in progress : Lesly-Ann Daniel, Security analysis of binary code, 1/10/2018, Sébastien Bardin and Tamara Rezk
- PhD in progress : Adam Khayam, Semantics of Multitier Languages, 1/07/2019, Alan Schmitt and Tamara Rezk
- PhD in progress: Ignacio Tiraboschi, Analysis for IoT Security, Xavier Rival and Tamara Rezk

10.2.3 Juries

- Tamara Rezk was a jury member of the national Inria competition CRCN/ISFP 2020.
- Tamara Rezk was a Phd jury member (Rapporteur) of Sebastian Poeplau (supervisor: Aurélien Francillon), EURECOM, 2020.

10.3 Popularization

10.3.1 Internal or external Inria responsibilities

- Tamara Rezk was part of the Editorial Board of the blog Binaire of Le Monde.

11 Scientific production

11.1 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: *\sc 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 \sc 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.

11.2 Publications of the year

International journals

[11] I. Castellani, M. Dezani-Ciancaglini, P. Giannini and R. Horne. 'Global types with internal delegation'. In: *Theoretical Computer Science* 807 (6th Feb. 2020), p. 26. DOI: 10.1016/j.tcs.2019.09.027. URL: https://hal.inria.fr/hal-02419937.

[12] B. Petit and M. Serrano. 'Skini:Reactive Programming fo Interactive Structured Music'. In: *The Art, Science, and Engineering of Programming* (8th June 2020). URL: https://hal.archives-ouvertes.fr/hal-03105643.

International peer-reviewed conferences

[13] I. Bastys, M. Balliu, T. Rezk and A. Sabelfeld. 'Clockwork: Tracking Remote Timing Attacks'. In: In Proceedings of the IEEE Computer Security Foundations Symposium (CSF). Virtual, France, 22nd June 2020. URL: https://hal.inria.fr/hal-03141411.

[14] G. Berry and M. Serrano. 'HipHop.js: (A)Synchronous reactive web programming'. In: PLDI '20 - 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation. London UK, United Kingdom, 15th July 2020, pp. 533–545. DOI: 10.1145/3385412.3385984. URL: https://hal.inria.fr/hal-03047902.

[15] M. El Laz, B. Grégoire and T. Rezk. 'Security Analysis of ElGamal Implementations'. In: 17th International Conference on Security and Cryptography. Lieusaint - Paris, France, 8th July 2020, pp. 310–321. DOI: 10.5220/0009817103100321. URL: https://hal.inria.fr/hal-03141511.

[16] M. Serrano and R. B. Findler. 'Dynamic property caches: a step towards faster JavaScript proxy objects'. In: CC '20 - 29th International Conference on Compiler Construction. San Diego CA, United States, 22nd Feb. 2020, pp. 108–118. DOI: 10.1145/3377555.3377888. URL: https://hal.inria.fr/hal-03047893.

Conferences without proceedings

[17] B. Petit and M. Serrano. 'Generative Music Using Reactive Programming'. In: International Computer Music Conférence. Santiago, Chile, 21st July 2021. URL: https://hal.archives-ouvertes.fr/hal-03105666.

Doctoral dissertations and habilitation theses

[18] B. Petit. 'Time and duration : from synchronous reactive programming to music composition'. Université Côte d'Azur, 2nd July 2020. URL: https://tel.archives-ouvertes.fr/tel-03135288.

Reports & preprints

[19] A. Canteaut, M. A. Fernández, L. Maranget, S. Perin, M. Ricchiuto, M. Serrano and E. Thomé. *Évaluation des Logiciels*. Inria, 14th Jan. 2021. URL: https://hal.inria.fr/hal-03110723.

[20] A. Canteaut, M. A. Fernández, L. Maranget, S. Perin, M. Ricchiuto, M. Serrano and E. Thomé. *Software Evaluation*. Inria, 14th Jan. 2021. URL: https://hal.inria.fr/hal-03110728.

[21] I. Castellani, M. Dezani-Ciancaglini and P. Giannini. *Global types and event structure semantics for asynchronous multiparty sessions*. 1st Feb. 2021. URL: https://hal.inria.fr/hal-03126627.