Activity Report 2014

Project-Team INDES

Secure Diffuse Programming

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Distributed programming and Software engineering

Table of contents

Project-Team INDES

Keywords: Programming Languages, Compiling, Security, Concurrency, Web

Creation of the Team: 2009 January 01*, updated into Project-Team:* 2014 November 26.

1. Members

Research Scientists

Manuel Serrano [Team leader, Inria, Senior Researcher, HdR]
Nataliia Bielova [Inria, Researcher]
Ilaria Castellani [Inria, Researcher]
Tamara Rezk [Inria, Researcher]
Bernard Serpette [Inria, Researcher]

Engineer

Vincent Prunet [Inria]

PhD Students

Yoann Couillec [Inria]
Johan Grande [Univ. Nice, until Sep 2014]
Cyprien Nicolas [Univ. Nice, until Aug 2014]
José Fragoso Santos [Univ. Nice, until Nov 2014]

Visiting Scientists

Atuya Okudaira [Professor, until Aug 2014]
Vineet Rajani [PhD student, from Dec 2014]

Administrative Assistant

Nathalie Bellesso [Inria]

Others

Gérard Boudol [Emeritus Researcher]
Diana Ioana Proteasa Nicola [M2 intern, from Mar 2014]

2. Overall Objectives

2.1. 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 and functional programming

We are studying new paradigms for programming Web applications that rely on multi-tier functional programming [6]. 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 [7]. 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.

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 [2], [1]. 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 [4]. 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 programming

Along with games, multimedia applications, electronic commerce, and email, the web has popularized computers for daily life. The revolution is engaged and we may be at the dawn of a new era of computing where the web is a central element. The web constitutes an infrastructure more versatile, polymorphic, and open, in other words, more powerful, than any dedicated network previously invented. For this very reason, it is likely that most of the computer programs we will write in the future, for professional purposes as well as for our own needs, will extensively rely on the web. In addition to allowing reactive and graphically pleasing interfaces, web applications are de facto distributed. Implementing an application with a web interface makes it instantly open to the world and accessible from much more than one computer. The web also partially solves the problem of platform compatibility because it physically separates the rendering engine from the computation engine. Therefore, the client does not have to make assumptions on the server hardware configuration, and vice versa. Lastly, HTML is highly durable. While traditional graphical toolkits evolve continuously, making existing interfaces obsolete and breaking backward compatibility, modern web browsers that render on the edge web pages are still able to correctly display the web pages of the early 1990?s. For these reasons, the web is arguably ready to escape the beaten track of n-tier applications, CGI scripting and interaction based on HTML forms. However, we think that it still lacks programming abstractions that minimize the overwhelming amount of technologies that need to be mastered when web programming is involved. Our experience on reactive and functional programming is used for bridging this gap.

4.2. Multimedia

Electronic equipments are less and less expensive and more and more widely spread out. Nowadays, in industrial countries, computers are almost as popular as TV sets. Today, almost everybody owns a mobile phone. Many are equipped with a GPS or a PDA. Modem, routers, NASes and other network appliances are also commonly used, although they are sometimes sealed under proprietary packaging such as the Livebox or the Freebox. Most of us evolve in an electronic environment which is rich but which is also populated with mostly isolated devices. The first multimedia applications on the web have appeared with the Web 2.0. The most famous ones are Flickr, YouTube, or Deezer. All these applications rely on the same principle: they allow roaming users to access the various multimedia resources available all over the Internet via their web browser. The convergence between our new electronic environment and the multimedia facilities offered by the web will allow engineers to create new applications. However, since these applications are complex to implement this will not happen until appropriate languages and tools are available. In the Indes team, we develop compilers, systems, and libraries that address this problem.

4.3. Robotics

The web is the de facto standard of communication for heterogeneous devices. The number of devices able to access the web is permanently increasing. Nowadays, even our mobile phones can access the web. Tomorrow it could even be the turn of our wristwatches! The web hence constitutes a compelling architecture for developing applications relying on the ambient computing facilities. However, since current programming languages do not allow us to develop easily these applications, ambient computing is currently based on ad-hoc solutions. Programming ambient computing via the web is still to be explored. The tools developed in the Indes team allow us to build prototypes of a robot as a web entity, and the use of remote web services to manage, monitor or extend the features of the robot. Among the direct benefits of relying on a web framework for robotics are the ability to use any web enabled device such as a smartphone or tablet to drive the robot.

5. New Software and Platforms

5.1. Introduction

Most INDES software packages, even the older stable ones that are not described in the following sections, are freely available on the Web. In particular, some are available directly from the Inria web site:

http://www.inria.fr/valorisation/logiciels/langages.fr.html

Most software packages can be downloaded from the INDES web site:

http://www-sop.inria.fr/teams/indes

5.2. Language-based Security

Participants: José Fragoso Santos, Tamara Rezk [correspondant].

5.2.1. JavaScript Library iflowtypes.js

The JavaScript library iflowtypes.js is designed to type secure information flow in JavaScript. iflowtypes.js has two main modes of operation: fully static and hybrid. In the hybrid mode, the program to be typed is instrumented with runtime assertions that are verified at runtime. By deferring rejection to runtime, the hybrid type system is able to type more programs than fully static mechanisms. This library is available at the URL: http://j3fsantos.github.io/PersonalPage/TypeSystem/.

5.2.2. JavaScript Library iflowsigs.js

The JavaScript library iflowsigs.js is designed to inline an information flow monitor into JavaScript code. iflowsigs.js supports is able to track information flow even in programs that interact with arbitrary Web APIs. This library is available at the URL: http://j3fsantos.github.io/PersonalPage/IFMonitor/.

5.3. Web programming

Participants: Yoann Couillec, Vincent Prunet, Manuel Serrano [correspondant].

5.3.1. The HOP web programming environment

HOP is a higher-order language designed for programming interactive web applications such as web agendas, web galleries, music players, etc. It exposes a programming model based on two computation levels. The first one is in charge of executing the logic of an application while the second one is in charge of executing the graphical user interface. HOP separates the logic and the graphical user interface but it packages them together and it supports strong collaboration between the two engines. The two execution flows communicate through function calls and event loops. Both ends can initiate communications.

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.

HOP has won the software *open source contest* organized by the ACM Multimedia Conference 2007. It is released under the GPL license. It is available at http://hop.inria.fr.

5.4. Old software

5.4.1. Camloo

Camloo is a caml-light to bigloo compiler, which was developed a few years ago to target bigloo 1.6c. New major releases 0.4.x of camloo have been done to support bigloo 3.4 and bigloo 3.5. Camloo make it possible for the user to develop seamlessly a multi-language project, where some files are written in caml-light, in C, and in bigloo. Unlike the previous versions of camloo, 0.4.x versions do not need a modified bigloo compiler to obtain good performance. Currently, the only supported backend for camloo is bigloo/C. We are currently rewriting the runtime of camloo in bigloo to get more portability and to be able to use HOP and camloo together.

5.4.2. Skribe

SKRIBE is a functional programming language designed for authoring documents, such as Web pages or technical reports. It is built on top of the SCHEME programming language. Its concrete syntax is simple and looks familiar to anyone used to markup languages. Authoring a document with SKRIBE is as simple as with HTML or LaTeX. It is even possible to use it without noticing that it is a programming language because of the conciseness of its original syntax: the ratio *tag/text* is smaller than with the other markup systems we have tested.

Executing a SKRIBE program with a SKRIBE evaluator produces a target document. It can be HTML files for Web browsers, a LaTeX file for high-quality printed documents, or a set of *info* pages for on-line documentation.

5.4.3. Scheme2JS

Scm2JS is a Scheme to JavaScript compiler distributed under the GPL license. Even though much effort has been spent on being as close as possible to R5RS, we concentrated mainly on efficiency and interoperability. Usually Scm2JS produces JavaScript code that is comparable (in speed) to hand-written code. In order to achieve this performance, Scm2JS is not completely R5RS compliant. In particular it lacks exact numbers.

Interoperability with existing JavaScript code is ensured by a JavaScript-like dot-notation to access JavaScript objects and by a flexible symbol-resolution implementation.

Scm2JS is used on a daily basis within HOP, where it generates the code which is sent to the clients (web-browsers). Scm2JS can be found at http://www-sop.inria.fr/indes/scheme2js.

5.4.4. The FunLoft language

FunLoft (described in http://www-sop.inria.fr/teams/indes/rp/FunLoft) is a programming language in which the focus is put on safety and multicore.

FunLoft is built on the model of FairThreads which makes concurrent programming simpler than usual preemptive-based techniques by providing a framework with a clear and sound semantics. FunLoft is designed with the following objectives:

- provide a safe language, in which, for example, data-races are impossible.
- control the use of resources (CPU and memory), for example, memory leaks cannot occur in FunLoft programs, which always react in finite time.
- have an efficient implementation which can deal with large numbers of concurrent components.
- benefit from the real parallelism offered by multicore machines.

A first experimental version of the compiler is available on the Reactive Programming site http://www-sop.inria.fr/teams/indes/rp. Several benchmarks are given, including cellular automata and simulation of colliding particles.

5.4.5. The Bigloo compiler

The programming environment for the Bigloo compiler [7] is available on the Inria Web site at the following URL: http://www-sop.inria.fr/teams/indes/fp/Bigloo. The distribution contains an optimizing compiler that delivers native code, JVM bytecode, and .NET CLR bytecode. It contains a debugger, a profiler, and various Bigloo development tools. The distribution also contains several user libraries that enable the implementation of realistic applications.

BIGLOO was initially designed for implementing compact stand-alone applications under Unix. Nowadays, it runs harmoniously under Linux and MacOSX. The effort initiated in 2002 for porting it to Microsoft Windows is pursued by external contributors. In addition to the native back-ends, the BIGLOO JVM back-end has enabled a new set of applications: Web services, Web browser plug-ins, cross platform development, etc. The new BIGLOO .NET CLR back-end that is fully operational since release 2.6e enables a smooth integration of Bigloo programs under the Microsoft .NET environment.

5.4.6. CFlow

The prototype compiler "CFlow" takes as input code annotated with information flow security labels for integrity and confidentiality and compiles to F# code that implements cryptography and protocols that satisfy the given security specification.

Cflow has been coded in F#, developed mainly on Linux using mono (as a substitute to .NET), and partially tested under Windows (relying on .NET and Cygwin). The code is distributed under the terms of the CeCILL-B license.

5.4.7. FHE type-checker

We have developed a type checker for programs that feature modern cryptographic primitives such as fully homomorphic encryption. The type checker is thought as an extension of the "CFlow" compiler developed last year on the same project. It is implemented in F#. The code is distributed under the terms of the CeCILL-B license.

5.4.8. Mashic compiler

The Mashic compiler is applied to mashups with untrusted scripts. The compiler generates mashups with sandboxed scripts, secured by the same origin policy of the browsers. The compiler is written in Bigloo and can be found at http://www-sop.inria.fr/indes/mashic/.

5.4.9. IFJS compiler

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.

The compiler is written in JavaScript and can be found at http://www-sop.inria.fr/indes/ifJS.

6. New Results

6.1. Web programming

Participants: Yoann Couillec, Vincent Prunet, Tamara Rezk, Manuel Serrano [correspondant].

6.1.1. Hop.js

Multitier programming languages unify within a single formalism and a single execution environment the programming of the different tiers of distributed applications. On the Web, this programming paradigm unifies the client tier, the server tier, and, when one is used, the database tier. This homogenization offers several advantages over traditional Web programming that rely on different languages and different environments for the two or three tiers of the Web application: programmers have only one language to learn, maintenance and evolution are simplified by the use of a single formalism, global static analyses are doable as a single semantics is involved, debugging and other runtime tools are more powerful as they access global informations about the execution [17].

The three first multitier platforms for the Web all appeared in 2006: GWT (a.k.a., Google Web Toolkit), Links, and Hop [6], [5]. Each relied on a different programming model and languages. GWT maps the Java programming model on the Web, as it allows, Java/Swing likes programs to be compiled and executed on the Web; Links is functional language with experimental features such as the storing of the whole execution context on the client; Hop is based on the Scheme programming language. These three pioneers have open the path for the other multitier languages such as, Ocsigen for Ocaml, UrWeb, js-scala, etc.

In spite of their interesting properties, multitier languages have not become that popular on the Web. Today, only GWT is widely used in industrial applications but arguably GWT is not a fully multitier language as developing applications with GWT requires explicit JavaScript and HTML programming. This lack of popularity of other systems is likely due to their core based languages than to the programming model itself.

JavaScript is the *defacto* standard on the Web. Since the mid 90's, it is the language of the client-side programming and more recently, with systems like nodejs, it is also a viable solution for the server-side programming. As we are convinced by the virtues of multitier programming we have started a new project consisting of enabling multitier programming JavaScript. We have created a new language called HopScript, which is a minimalist extension of JavaScript for multitier programming, and we have implemented a brand new runtime environment called Hop.js. This environment contains a builtin Web server, on-the-fly HopScript compilers, and many runtime libraries.

HopScript is a super set of JavaScript, *i.e.,* all JavaScript programs are legal HopScript programs. Hop.js is a compliant JavaScript execution environment as it succeeds at 99% of the Ecma 262 tests suite. The Hop.js environment also aims at Node.js compatibility. In its current version it supports about 70% of the Node.js runtime environment. In particular, it fully supports the Node.js modules, which lets Hop programs reuse existing Node.js modules as is.

A prototype version of Hop.js is currently used by several academic and SME R&D teams to jointly develop an assistive robotic platform and a set of distributed applications.

We plan to release the first public Hop.js version by the end of the first semester of 2015, as we plan to start describing in forthcoming papers.

6.1.2. Multitier Debugging

Debugging Web applications is difficult because of their distributed nature and because the server-side and the client-side of the application are generally treated separately. The multitier approach, which reunifies the two ends of the application inside a unique execution environment, helps the debugging process because it lets the debugger access more runtime informations.

Based on our previous work on the Hop multitier debugger [17], we have built a multitier debugger for Hop.js, our multitier extension of JavaScript. Its advantage over most debuggers for the Web is that it reports the full stack trace containing all the server-side and client-side frames that have conducted to an error. Errors are reported on their actual position on the source code, wherever they occur on the server or on the client. This paper presents this debugger and sketches its implementation. This work is described in a yet unpublished paper, which will appear in 2015.

6.1.3. Datasource

We extended the HOP.JS language with an embedded language, inspired by PLINQ and ORC, called DATASOURCE. It allows programmers request multiple data sources with queries written in a unique language. We used a plinq-like language to express queries and an orc-like language to orchestrate them. Our query language and the orchastration languages can be used simultaneously or separately. We implemented bindings between DATASOURCE and some representative types of data sets such as SPARQL endpoints, relationnal databases, WEB services, and WEB pages. We are extending HOP.JS by supporting EcmaScript 6 array comprehensions in order to write a unique query over multiple data sources in a unified formalism. The query is then compiled into database specific queries. We linked all the bindings made for HOP with HOP.JS. We implemented another binding for a document oriented data base, MONGODB.

6.2. Distributed programming

Participant: Bernard Serpette [correspondant].

6.2.1. Logical behavioural semantics of Esterel

We have formalised, with the Coq system, the logical behavioural semantics of Esterel as described in Gérard Berry's book. In order to define the properties of reactivity and determinism, we have defined a new semantics using contexts with a proven correspondence between the two semantics.

The specification and the proofs of the correspondence take 3500 lines of Coq.

6.2.2. Abstract distributed machine

We have experimented an abstract machine composed of distributed nodes. Each node has exactly two named links to other nodes and an instruction able to modify one link of a reachable node. This instruction is executed when a token is received, once the instruction is achieved the token is transmitted to another reachable node.

This abstract machine is turing complete. The λ-calculus and the π-calculus can be compiled to the instruction set of this machine.

The execution of one individual node may involve paths of arbitrary length, for example, when compiling the λ-calculus or the π-calculus, the path length for accessing a variable is proportional to it's de Bruijn index and therefore is not bounded. Given a machine with instructions of unbounded paths, we can build an equivalent machine where all the paths are bounded by two: a node is only able to access it's own links and the links of its neighbour. Moreover, this transformation uses only 6 different instructions.

6.3. Security and Privacy

Participants: Ilaria Castellani, José Fragoso Santos, Nataliia Bielova, Tamara Rezk [correspondant].

6.3.1. Security of Dynamically Evolving Systems of Communicating Processes

We have started to address security issues in the context of dynamically evolving systems of communicating processes, which are able to adapt themselves in reaction to particular events (for instance, security attacks or changes in security policies). We present initial results on a simple model of processes communicating via structured interactions (sessions), in which self-adaptation and security concerns are jointly addressed. In this model, security violations occur when processes attempt to read or write messages of inappropriate security level within a structured interaction. Such violations trigger adaptation mechanisms that prevent the violations to occur and/or to propagate their effect in the choreography. Our model is equipped with local and global mechanisms for reacting to security violations; type soundness results ensure that the global protocols are still correctly executed while the system adapts itself to preserve its security.

6.3.2. Browser Randomisation against Fingerprinting: a Quantitative Information Flow Approach

Web tracking companies use device fingerprinting to distinguish the users of the websites by checking the numerous properties of their machines and web browsers. One way to protect the users' privacy is to make them switch between different machine and browser configurations. We propose a formalisation of this privacy enforcement mechanism.

We use information-theoretic channels to model the knowledge of the tracker and the fingerprinting program, and show how to synthesise a randomisation mechanism that defines the distribution of configurations for each user. This mechanism provides a strong guarantee of *privacy* (the probability of identifying the user is bounded by a given threshold) while maximising *usability* (the user switches to other configurations rarely). To find an optimal solution, we express the enforcement problem of randomisation by a linear program. We investigate and compare several approaches to randomisation and find that more efficient privacy enforcement would often provide lower usability. Finally, we relax the requirement of knowing the fingerprinting program in advance, by proposing a randomisation mechanism that guarantees privacy for an arbitrary program.

This work has been published and presented at the Nordic Conference on Secure IT Systems (NordSec 2014) [12]. The extended version of the paper has been published as a technical report [20].

6.3.3. Crying Wolf? On the Price Discrimination of Online Airline Tickets

Price discrimination refers to the practice of dynamically varying the prices of goods based on a customer's purchasing power and willingness to pay. Motivated by several anecdotal accounts, we report on a three week experiment, conducted in search of price discrimination in airline tickets. Despite presenting the companies with multiple opportunities for discriminating us, and contrary to our expectations, we did not find any evidence for systematic price discrimination. At the same time, we witnessed the highly volatile prices of certain airlines which make it hard to establish cause and effect. Finally, we provided alternative explanations for the observed price differences.

This work has been published and presented at the Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014) [19].

6.3.4. Stateful Declassification Policies for Event-Driven Programs

We propose a novel mechanism for enforcing information flow policies with support for declassification on event-driven programs. Declassification policies consist of two functions. First, a projection function specifies for each confidential event what information in the event can be declassified directly. This generalizes the traditional security labelling of inputs. Second, a stateful release function specifies the aggregate information about all confidential events seen so far that can be declassified. We provide evidence that such declassification policies are useful in the context of JavaScript web applications. An enforcement mechanism for our policies is presented and its soundness and precision is proven. Finally, we give evidence of practicality by implementing and evaluating the mechanism in a browser. This work has been published at Computer Security Foundations (CSF'14) [18].

6.3.5. An Information Flow Monitor for a Core of DOM

We propose and prove sound a novel, purely dynamic, flow sensitive monitor for securing information flow in an imperative language extended with DOM-like tree operations, that we call Core DOM. In Core DOM, as in the DOM API, tree nodes are treated as first-class values. We take advantage of this feature in order to implement an information flow control mechanism that is finer-grained than previous approaches in the literature. Furthermore, we extend Core DOM with additional constructs to model the behavior of live collections in the DOM Core Level 1 API. We show that this kind of construct effectively augments the observational power of an attacker and we modify the proposed monitor so as to tackle newly introduced forms of information leaks. This work has been published at the 9th International Symposium on Trustworthy Global Computing (TGC) [11].

6.3.6. An Information Flow Monitor-Inlining Compiler for Securing a Core of JavaScript

Web application designers and users alike are interested in isolation properties for trusted JavaScript code in order to prevent confidential resources from being leaked to untrusted parties. Noninterference provides the mathematical foundation for reasoning precisely about the information flows that take place during the execution of a program. Due to the dynamicity of the language, research on mechanisms for enforcing noninterference in JavaScript has mostly focused on dynamic approaches. We present the first information flow monitor inlining compiler for a realistic core of JavaScript. We prove that the proposed compiler enforces termination-insensitive noninterference and we provide an implementation that illustrates its applicability.

This work has been published at the 29th IFIP International Information Security and Privacy Conference (IFIP SEC) [14].

6.3.7. From Static to Hybrid Typing Secure Information Flow in a Core of JavaScript

We propose a novel type system for securing information flow in a core of JavaScript. This core takes into account the defining features of the language, such as prototypical inheritance, extensible objects, and constructs that check the existence of object properties. We design a hybrid version of the proposed type system. This version infers a set of assertions under which a program can be securely accepted and instruments it so as to dynamically check whether these assertions hold. By deferring rejection to runtime, the hybrid version can typecheck secure programs that purely static type systems cannot accept.

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. ANR DEFIS PWD

The PWD project (Programmation du Web diffus) has been funded by the ANR Défis programme for 4 years, starting November 2009. The partners of this project are the teams INDES (coordinator), LIP6 at University Pierre et Marie Curie and PPS at University Denis Diderot. The PWD project has been elected as one the projects "phare" by the ANR.

7.1.2. ANR AJACS

The AJACS project (Analyses of JavaScript Applications: Certification & Security) has been funded by the ANR for 42 months, starting December 2014. The goal of AJACS project is to provide strong security and privacy guarantees on the client side for web application scripts. The Indes members are involved in the tasks WP2 Certified Analyses and WP3 Security of JavaScript Applications. The partners of this project include Inria teams Celtique (coordinator), Toccata, and Prosecco.

7.1.3. FUI X-Data

Broadly available big and open data open new perspectives in terms of use and applications. The X-Data project aims at validating this claim by using actual data sets for building realistic applications. The goal is to combine a large variety of data sets coming from different partners (Data Publica, Orange, EDF, La Poste, social networks, ...) to build innovative applications. The Indes team designs and implements new programming language constructs that help programming these applications. Our contribution to this project ended in November 2014.

7.1.4. FUI UCF

The 3 years long UCF project aims at developing a reactive Web platforms for delivering multimedia contents. The partners of the project are the startups Alterway, OCamlPro, and XWiki, and the academic research laboratories of University Pierre et Marie Curie and Denis Diderot.

7.2. European Initiatives

7.2.1. FP7

7.2.1.1. RAPP

Program: http://rapp-project.eu

Title: Robot App Store

Collaborator: Inria Hephaistos

Abstract: RAPP is a 36 months pan-european FP7 project, started in December 2013. Hop is used in the development of prototypes of the Coprin Ang rollator transfer device, for mobility assistance and activity monitoring.

7.2.1.2. MEALS

Type: FP7

Title: Mobility between Europe and Argentina applying Logics to Systems

Instrument: International Research Staff Exchange Scheme

Duration: October 2011 - September 2015

Coordinator: Pedro D'Argenio

Partner: University of Córdoba, University of Buenos Aires, University of Twente

Inria contact: Castuscia Palamidessi

Abstract:The MEALS project (Mobility between Europe and Argentina applying Logics to Systems) goals cover three aspects of formal methods: specification (of both requirement properties and system behavior), verification, and synthesis. The Indes members are involved in the task of Security and Information Flow Properties (WP3). The partners in this task include University of Buenos Aires, University of Córdoba, Inria (together with Catuscia Palamidessi, Kostas Chatzikokolakis, Miguel Andrés) and University of Twente. The web page of the project can be found at http://www.meals-project.eu.

7.2.2. Collaborations in European Programs, except FP7 & H2020

Program: **ICT Cost Action IC1201**

Project acronym: BETTY

Project title: Behavioural Types for Reliable Large-Scale Software Systems

Duration: October 2012 - October 2016

Coordinator: Simon Gay, University of Glasgow

Other partners: Several research groups, belonging to 22 european countries

Abstract: The aim of BETTY is to investigate and promote behavioural type theory as the basis for new foundations, programming languages, and software development methods for communication-intensive distributed systems. Behavioural type theory encompasses concepts such as interfaces, communication protocols, contracts, and choreography.

Program: **ICT Cost Action IC1405**

Project title: Reversible computation - extending horizons of computing

Duration: November 2014 - November 2018

Coordinator: Irek Ulidowski, University of Leicester

Abstract: Reversible computation is an emerging paradigm that extends the standard forwards mode of computation with the ability to execute in reverse. It aims to deliver novel computing devices and software, and to enhance traditional systems. The potential benefits include the design of reversible logic gates and circuits - leading to low-power computing and innovative hardware for green ICT, new conceptual frameworks and language abstractions, and software tools for reliable and recovery-oriented distributed systems.

This Action is the first European network of excellence aimed at coordinating research on reversible computation.

7.3. International Research Visitors

7.3.1. Visits of International Scientists

7.3.1.1. Internships

Vineet Rajani

Date: 10/12/2014 - 10/03/2015

Institution: Max Planck Institute (MPI), Germany

Collaborator: Tamara Rezk

7.3.1.2. Research stays abroad

Atuya Okudaira

Date: 1/1/2014 - 31/08/2014

Institution: International University of Kagoshima, Japan

Collaborator: Manuel Serrano

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific events organisation

8.1.1.1. general chair, scientific chair

- **Ilaria Castellani** is the chair of the IFIP WG 1.8 on Concurrency Theory (of which she has been a member since its start in 2005). She is a member of the Management Committee of the european COST Action IC1201 BETTY on Behavioural Types, and the chair of the BETTY working group on Security. Since November 2014 she is also a member of the COST Action IC1405 on Reversible Computation.
- **Manuel Serrano** is the coordinator of the ANR DEFIS project PWD.
- **Tamara Rezk** is the coordinator of the security work package (WP3) of the ANR AJACS project.

8.1.1.2. member of the organizing committee

- **Ilaria Castellani** co-organised (with the WG Secretary Mohammad Reza Mousavi) the workshop TRENDS 2014 and the annual business meeting of WG 1.8, and she participated in the IFIP TC1 (Foundations of Computation) business meeting, all held in September in Rome. She was involved in the organisation of the OPCT (Open Problems in Concurrency Theory) workshop that took place in May in Bertinoro, which was co-sponsored by WG 1.8. She was also one of the organisers of the MatthewFest, a two-day workshop that took place in Lucca in October to celebrate the 65th anniversary of Matthew Hennessy. She reported on the 2013 workshop ?25 Years of Combining Compositionality and Concurrency" (WS25CCC) in the Bulletin of EATCS.

- **Tamara Rezk** organized a workshop on Programming Languages and Verification and a workshop on JavaScript Security at Inria Sophia Antipolis, on 9th and 16th December 2014.

8.1.2. Scientific events selection

8.1.2.1. member of the conference program committee

- **Manuel Serrano** served on the program committee of the *16th Practical Aspects of Declarative Languages* (PADL'14) conference, *Trends in Functional Programming*, (TFP'14) and *Web Audio Conference* (WAC'15).
- **Nataliia Bielova** was a publication chair of International Symposium on Engineering Secure Software and Systems (ESSoS 2014) and was invited to be a publication chair of ESSoS 2015.
- **Ilaria Castellani** was a member of the programme committee of the workshop BEAT 2014.
- **Tamara Rezk** served on the program committees of Computer Security Foundations (CSF'14), of International Symposium on Engineering Secure Software and Systems (ESSOS'15), and of 4th Conference on Principles of Security and Trust (POST'15).

8.1.2.2. reviewer

- **Nataliia Bielova** served as an external reviewer to the Conference on Principles of Security and Trust (POST 2014) and Computer Security Foundations Symposium (CSF 2014).

8.1.3. Journal

8.1.3.1. member of the editorial board

- **Ilaria Castellani** is a member of the editorial board of *Technique et Science Informatiques*.

8.1.3.2. reviewer

- **Nataliia Bielova** was an invited reviewer of the ACM Transactions on Information and System Security (TISSEC).
- **Tamara Rezk** was an invited reviewer of the Journal Computer Languages, Systems & Structures.

8.2. Seminars and conferences

- **Manuel Serrano** gave two seminars about Web programming in various events: "Future of Programming", Delft 2014 http://eelcovisser.org/wiki/future-of-programming/program and "9th Symposium on Future Trends in Service-Oriented Computing", Potsdam 2014. He participated in the SAC'2014 conference in Gyeongju, Korea [16], where is presented new techniques for implementing locks efficiently in higher order programming languages. He also participated in the WEBIST'14 conference in Barcelona, Spain [17], where he presented his work on multitier debugging of Web applications.
- **Bernard Serpette** gave a talk about *Unification des couleurs dans un λ-calcul polychrome* at the JFLA'14, Fréjus, France.
- **Nataliia Bielova** was invited to present her work on browser randomisation against fingerprinting at PRINCESS workshop in December 2014.
- **Ilaria Castellani** participated in the NII (the japanese National Institute of Informatics) Shonan Meeting on "Software Contracts for Communication, Monitoring, and Security", held at Shonan Village Center. In June she took part in the workshop OPCT (Open Problems in Concurrency Theory) in Bertinoro, Italy. In both cases, she gave a talk on Security for Reactive Synchronous Languages. In September, she participated in the conference TCS (Theoretical Computer Science) in Rome, where she gave a joint talk with Marco Bernardo, reporting on the IFIP WG 1.8 activities and on the OPCT workshop, in a special session dedicated to the IFIP TC1 working groups. She also attended the BEAT workshop, where her work was presented by Jorge A. Perez. She finally participated, as a co-organiser, in the workshop TRENDS 2014 and in the event MatthewFest (cf Section 9.2 below). As part of the Action BETTY, she participated in two one-day project meetings/workshops, associated with the conferences ETAPS and CONCUR respectively, and she spent a short visit to the University of Torino and another one to Trinity College, Dublin.

- **José Fragoso Santos** participated in the 29th IFIP International Conference of Information Security and Privacy (IFIP SEC'2014) in which he presented his joint work with Tamara Rezk (An Information Flow Monitor-Inlining Compiler for Securing a Core of JavaScript).

  He participated in the 9th International Symposium on Trustworthy Global Computing (TGC'2014) in which he presented his joint work with Ana Almeida Matos and Tamara Rezk (An Information Flow Monitor for a Core of DOM - Introducing References and Live Primitives).

  He gave a talk about information flow security for client-side Web applications at the LoReL group and at the LaFHIS group in the University of Buenos Aires and at the Dependable Systems Group in the University of Córdoba. José Fragoso Santos also gave a talk about information flow security for the DOM API in the Department of Computing of the Imperial College in London.
- **Tamara Rezk** was invited to Dagstuhl Seminar 14271, "Scripting Languages and Frameworks: Analysis and Verification". She gave two talks on "Hybrid Typing for JavaScript" and "Hop Operational Semantics".

8.3. Teaching - Supervision - Juries

8.3.1. Teaching

Licence : **Vincent Prunet**, *Algorithms and Data Structures*, 80 ETD, L2, Lycée International de Valbonne Sophia Antipolis (within the scope of the national Inria action to promote early CS courses in all scientific curricula), France.

Master : **Manuel Serrano**, *Programming the Diffuse Web*, 26h ETD, M2, University Paris 6 (UPMC), France.

Master : **Tamara Rezk**, *Security of Web Applications*, 36 ETD, M2, University of Nice Sophia Antipolis, France.

Master : **Tamara Rezk**, *Provable Cryptography*, 36 ETD, M2, University of Nice Sophia Antipolis, France.

Master: **José Fragoso Santos**, *Software Security*, 6 ETD, M2, of Instituto Superior Técnico University of Lisbon, Portugal.

Doctorat : **Manuel Serrano**, Hop.js, full-day seminar, *École des Jeunes Chercheurs en Programmation* ,Rennes, France.

Doctorat: **Ilaria Castellani**, Behavioural types, one-hour tutorial, Lovran School.

8.3.2. Supervision

PhD : **José Fragoso Santos**, *Enforcing Secure Information Flow in Client Side Web Applications*, University of Nice Sophia Antipolis, 08/12/2014, **Ana Almeida Matos** and **Tamara Rezk**.

PhD in progress : **Cyprien Nicolas**, *Orchestrating multi-tier programming languages*, University of Nice Sophia Antipolis, 1/09/2010, **Gérard Berry** and **Manuel Serrano**.

PhD in progress: **Johan Grande**, *Conception et implantation d'un langage de programmation concurrente modulaire*, University of Nice Sophia Antipolis, 1/10/2010, **Gérard Boudol** and **Manuel Serrano**.

PhD in progress: **Yoann Couillec**, *Langages de programmation et données ouvertes*, University of Nice Sophia Antipolis, 1/10/2012, **Manuel Serrano** and **Patrick Valduriez**.

PFE in progress: **Julien Chiaremello**, *Décomposition en nombres premiers en informatique quantique*, University of Nice Sophia Antipolis, 1/11/2014, **Tamara Rezk**.

Master Internship: **Diana Proteasa-Nicola** on activity monitoring within Hop applications, Polytech Nice Sophia, **Vincent Prunet**.

Inria Internship Program: **Diana Proteasa-Nicola** on Robot simulation with Hop.js, Universitatea Politehnica Timisoara, **Vincent Prunet**.

8.3.3. Juries

- Manuel Serrano was a reviewer of the PhD thesis of Cagdas Bozman, ENSTA, Paris. He was also a reviewer of the PhD thesis of Julien Richard Roy, University of Rennes. Manuel Serrano organized and headed the CR2 jury of Inria Sophia-Antipolis.
- Ilaria Castellani was a reviewer of the PhD thesis of Ornela Dardha, University of Bologna.
- Tamara Rezk was a reviewer (rapporteur) of the PhD thesis of Carlos Luna, University of the República, Uruguay. He was also part of the jury of the PhD thesis of José Fragoso Santos, University of Nice Sophia Antipolis.

8.4. Transfer

8.4.1. WebRobotics

Dissemination of the HOP technology has become a priority for the team now that HOP is actually used to develop large projects. In 2012, a further step was taken with the allocation of dedicated resources missioned to develop and transfer the application portfolio to the industry. The team has focused on bringing web awareness to personal assistance robots developed by the Hephaistos team, also at Inria CRISAM, in line with one of the top strategic orientations of Inria. Using web protocols as a native framework greatly simplifies the integration of the robot as a web entity, and the use of remote web services to manage, monitor or extend the features of the robot. The behavior of a HOP robot is specified in HOP and orchestrated within diffuse HOP run time agents embedded within the robot elements, in charge of handling communication and control between platforms and with remote web services. The project, code-named *WebRobotics*, builds on the experience gained in using HOP for home automation over the recent years, adding in 2013 the support of versatile robotic computing platforms (integration with the ROS robot framework, development of a lightweight Hop client library providing support of web protocols to legacy software components). Among the direct benefits of relying on a web framework are the ability to use any web enabled device such as a smartphone or tablet to drive the robot. Also, it is much simpler to put in place remote diagnostic and monitoring services by leveraging on existing robot sensors and the HOP framework.

The *WebRobotics* project is now part of the RAPP FP7 european project, launched in December 2013, where Hop technology is used by several academic and SME R&D teams to develop a distributed software platform and applications for assistive robotics. Two prototypes are being developed, the first one is a personal coach robot (a Nao humanoid robot embedding Hop distributed applications), and the second one is a smart rollator (a walking aid with additional hardware and software services for rehabilitation, training and activity monitoring. The rollator hardware and robotic components are provided by Inria Hephaistos). In 2013, Indes has initiated a collaboration with other research teams (Inria STARS, Nice University Cobtek Project) and local institutes and SMEs to foster the development distributed monitoring and supervision applications with the Hop technology.

Indes has presented Hop applications for robots at the Lille Inria Industry conference day in November 2014.

9. Bibliography

Major publications by the team in recent years

[1] G. BARTHE, T. REZK, A. RUSSO, A. SABELFELD. *Security of Multithreaded Programs by Compilation*, in "ESORICS", 2007, pp. 2-18

[2] G. BOUDOL, I. CASTELLANI. *Noninterference for Concurrent Programs and Thread Systems*, in "Theoretical Computer Science", 2002, vol. 281, n^{o} 1, pp. 109-130

[3] G. BOUDOL, Z. LUO, T. REZK, M. SERRANO. *Reasoning about Web Applications: An Operational Semantics for HOP*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", 2012, vol. 34, n^{o} 2

[4] C. FOURNET, 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

[5] M. SERRANO, G. BERRY. *Multitier Programming in Hop - A first step toward programming 21st-century applications*, in "Communications of the ACM", August 2012, vol. 55, n° 8, pp. 53–59 [*DOI :* 10.1145/2240236.2240253], http://cacm.acm.org/magazines/2012/8/153796-multitier-programming-in-hop/abstract

[6] M. SERRANO, E. GALLESIO, F. LOITSCH. *HOP, a language for programming the Web 2.0*, in "Proceedings of the First Dynamic Languages Symposium", Portland, Oregon, USA, October 2006

[7] M. SERRANO. *Bee: an Integrated Development Environment for the Scheme Programming Language*, in "Journal of Functional Programming", May 2000, vol. 10, n° 2, pp. 1–43

Publications of the year

Doctoral Dissertations and Habilitation Theses

[8] J. FRAGOSO SANTOS. *Enforcing Secure Information Flow in Client-Side Web Applications*, University of Nice Sophia Antipolis, December 2014, https://hal.inria.fr/tel-01098548

Articles in International Peer-Reviewed Journals

[9] S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI. *Typing access control and secure information flow in sessions*, in "Journal of Information and Computation", 2014, vol. 238, pp. 68 - 105 [*DOI :* 10.1016/J.IC.2014.07.005], https://hal.inria.fr/hal-01088782

Invited Conferences

[10] G. BERRY, M. SERRANO. *Hop and HipHop : Multitier Web Orchestration*, in "International Conference on Distributed Computing and Internet Technology", Bhubaneswar, India, February 2014, https://hal.inria.fr/hal-00911782

International Conferences with Proceedings

[11] A. ALMEIDA MATOS, J. FRAGOSO SANTOS, T. REZK. *An Information Flow Monitor for a Core of DOM*, in "Symposium on Trustworthy Global Computing (TGC)", Rome, Italy, September 2014, https://hal.inria.fr/hal-01087375

[12] F. BESSON, N. BIELOVA, T. JENSEN. *Browser Randomisation against Fingerprinting: A Quantitative Information Flow Approach*, in "Nordic Conference on Secure IT Systems (NordSec)", Tromsø, Norway, October 2014 [*DOI :* 10.1007/978-3-319-11599-3_11], https://hal.inria.fr/hal-01081037

[13] I. CASTELLANI, M. DEZANI-CIANCAGLINI, J. A. PEREZ. *Self-Adaptation and Secure Information Flow in Multiparty Structured Communications: A Unified Perspective*, in "Third Workshop on Behavioural Types (BEAT)", Rome, Italy, Marco Carbone, September 2014, vol. 162, pp. 9 - 18 [*DOI :* 10.4204/EPTCS.162.2], https://hal.inria.fr/hal-01088437

[14] J. FRAGOSO SANTOS, T. REZK. *An Information Flow Monitor-Inlining Compiler for Securing a Core of JavaScript*, in "IFIP International Information Security and Privacy Conference (IFIP SEC)", Marrakesh, Morocco, June 2014, pp. 278 - 292 [*DOI :* 10.1007/978-3-642-55415-5_23], https://hal.inria.fr/hal-01087374

[15] F. PSOMOPOULOS, E. TSARDOULIAS, A. GIOKAS, C. ZIELINSKI, V. PRUNET, I. TROCHIDIS, D. DANEY, M. SERRANO, L. COURTES, S. ARAMPATZIS, P. A. MITKAS. *RAPP System Architecture*, in "Assistance and Service Robotics in a Human Environment, IEEE/RSJ International Conference on Intelligent Robots and Systems", chicago, United States, September 2014, https://hal.inria.fr/hal-01090891

[16] M. SERRANO, G. JOHAN. *Locking Fast*, in "Symposium on Applied Computing", Gyeongju, South Korea, ACM, March 2014, https://hal.inria.fr/hal-00912569

[17] M. SERRANO. *A Multitier Debugger for Web Applications*, in "WEBIST'14", Barcelone, Spain, April 2014, https://hal.inria.fr/hal-00980605

[18] M. VANHOEF, W. DE GROEF, D. DEVRIESE, F. PIESSENS, T. REZK. *Stateful Declassification Policies for Event-Driven Programs*, in "Computer Security Foundations (CSF'14)", Viena, Austria, July 2014, pp. 293 - 307 [*DOI :* 10.1109/CSF.2014.28], https://hal.inria.fr/hal-01098443

[19] T. VISSERS, N. NIKIFORAKIS, N. BIELOVA, W. JOOSEN. *Crying Wolf? On the Price Discrimination of Online Airline Tickets*, in "7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs)", Amsterdam, Netherlands, July 2014, https://hal.inria.fr/hal-01081034

Research Reports

[20] F. BESSON, N. BIELOVA, T. JENSEN. *Enforcing Browser Anonymity with Quantitative Information Flow*, 2014, n^{o} RR-8532, https://hal.inria.fr/hal-00984654