Activity Report 2016

Project-Team INDES

Secure Diffuse Programming

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Distributed programming and Software engineering

Table of contents

Project-Team INDES

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

Keywords:

Computer Science and Digital Science:

1.3. - Distributed Systems
2. - Software
2.1. - Programming Languages
2.1.3. - Functional programming
2.1.7. - Distributed programming
2.1.8. - Synchronous languages
2.1.9. - Dynamic languages
2.2.1. - Static analysis
2.2.3. - Run-time systems
4. - Security and privacy
4.3.3. - Cryptographic protocols
4.6. - Authentication
4.7. - Access control
4.8. - Privacy-enhancing technologies

Other Research Topics and Application Domains:

6.3.1. - Web
6.4. - Internet of things
9.4.1. - Computer science
9.8. - Privacy

1. Members

Research Scientists

Manuel Serrano [Team leader, Inria, Senior Researcher, HDR]
Nataliia Bielova [Inria, Researcher]
Gérard Boudol [Inria, Senior Researcher, until Jun 2016, HDR]
Ilaria Castellani [Inria, Researcher]
Tamara Rezk [Inria, Researcher]
Bernard Serpette [Inria, Researcher]

Engineers

Cédric Duminy [Inria]
Vincent Prunet [Inria, granted by FP7 RAPP project]

PhD Students

Yoann Couillec [Inria, until June 2016]
Dolière Francis Some [Inria]
Colin Vidal [Inria]

Post-Doctoral Fellow

Nguyen Nhat Minh Ngo [Inria, from October 2016]

Administrative Assistant

Nathalie Bellesso [Inria]

Other

Raimil Cruz Concepcion [University of Chile, Visiting Scientist, from May 2016 until July 2016]

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 [8]. 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 [9]. 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 [6]. 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. New Software and Platforms

4.1. Hop

KEYWORDS: Domotique - Web 2.0 - Iot - Functional language - Programming
SCIENTIFIC DESCRIPTION

Hop.js is a platform for web, cloud, and IoT applications. Its development environment is composed of:

- a programming language named HopScript, which is based on ECMAScript 262, //aka// JavaScript;
- an optimized web server;
- on-the-fly compilers for generating HTML, CSS, and client-side JavaScript;
- an ahead-of-time compiler for compiling JavaScript to native code;
- numerous APIs for networking, multimedia, robotics, IoT, etc.

The HopScript language extends JavaScript to consistently define the server and client part of web applications and IoT applications. HopScript supports syntactic forms that help creating HTML elements. It supports services that enable function calls over HTTP. Being at higher level than traditional Ajax programming, Hop.js services avoid the burden and pitfalls of URL management and explicit data marshalling. They combine the benefits of a high level RPC mechanism and low level HTTP compatibility.

Although Hop.js can be used to develop traditional web servers, it is particularly adapted to the development of web applications embedded into devices, where the server and client part of the application are intimately interoperating with each other. The programing model of Hop.js fosters the joint specification of server and client code, and allows the rapid development of web user interfaces, on the client, controlling the execution of the distributed application. By defining a single data model, providing functions that can run indifferently on both sides, and almost forgetting about client-server protocols, Hop.js seems well suited for agile development of web applications for this class of applications.

- Participants: Manuel Serrano and Vincent Prunet
- Contact: Manuel Serrano
- URL: http://hop.inria.fr

4.2. Mashic

FUNCTIONAL DESCRIPTION

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.

- Contact: Tamara Rezk
- URL: http://web.ist.utl.pt/~ana.matos/Mashic/mashic.html

4.3. Webstats

Webstats is a follow-up of the internship on JavaScript constructs used in top Alexa sites, started in summer 2015 by Dolière Francis Some. He analyzed the top 10,000 Alexa sites, and provided statistics about them. Among those statistics, their are:

- the most popular JavaScript libraries
- the most recurrent JavaScript constructs
- the adoption of security features such as:
  - The Content Security Policy, a policy for defending against Cross-Site-Script attacks
  - HttpOnly and Secure cookies, that prevents attacks like session hijacking.

Starting from April, 2016, this study is performed periodically, at the end of each month. The results are accessible online at https://webstats.inria.fr.

- Contact: Francis Some
- URL: https://webstats.inria.fr

5. New Results

5.1. Web programming

Participants: Cédric Duminy, Vincent Prunet, Bernard Serpette, Manuel Serrano [correspondant], Colin Vidal.

Hop.js [20], [22] is a new platform for web applications, potentially involving interconnected servers. The server-side execution is compatible with Node.js. Programmers then benefit from numerous existing libraries and applications. Hop.js also introduces distinctive programming features that are expressed in the HopScript programming language, a multitier extension of JavaScript. The Hop.js runtime embeds a multi-backends HopScript compiler.

The HopScript language extends JavaScript to consistently define the server and client part of a web application. HopScript supports syntactic forms that help creating HTML elements. It supports services that enable function calls over HTTP. Being at higher level than traditional Ajax programming, Hop.js services avoid the burden and pitfalls of URL management and explicit data marshalling. They combine the benefits of a high level RPC mechanism and low level HTTP compatibility.

Hop.js supports server-side and client-side parallelism. On the server, it first relies on its built-in pipelining architecture that automatically decodes HTTP requests in parallel. It also relies on server-side web workers that programs may explicitly launch to perform background tasks (functions and services). Each worker runs its own system thread. The service invocation and execution API fully integrates with the JavaScript execution flow, allowing synchronous and asynchronous operations on both client and server processes. The asynchronous response API can be combined with the worker API, allowing processing and asynchronous service responses to be delegated between workers. On the browser client-side parallelism relies on standard web workers.

Although Hop.js can be used to develop traditional web servers, it is particularly adapted to the development of web applications embedded into devices, where the server and client part of the application are intimately interoperating with each other. The programing model of Hop.js fosters the joint specification of server and client code, and allows the rapid development of web user interfaces, on the client, controlling the execution of the distributed application. By defining a single data model, providing functions that can run indifferently on both sides, and almost forgetting about client-server protocols, Hop.js seems well suited for agile development of web applications for this class of applications.

As an example, Hop.js has already been successfully used as the core framework to develop embedded and cloud applications for connected robots and IoT devices. In the context of a European industrial collaborative project, it has been used by various categories of programmers (mostly undergraduate internships, robotic experts, and professional engineers familiar with web development techniques) to build complex distributed applications, where various sort of digital equipments (computers, robots, small devices) communicate with each other, discover themselves, and collaborate. In all cases we have observed an easy adoption from everyone. The tons of JavaScript resources and examples available on the web helped internship students to rapidly become productive. Robotic experts were instantly able to start implementing Hop.js applications. Web experts seemed to feel at home with Hop.js as it let them build working applications with Hop.js core features and then extend them with existing JavaScript third party modules, typically npm modules.

In 2016, we first version of Hop.js as been completed and released. It is available from the Web site http://hop.inria.fr.

5.1.1. Web Reactive Programming

Web UI interfaces are specified as HTML documents. When instantiated in a browser these documents are accessible from JavaScript as abstract data structures conforming to the Document Object Model (*aka* the DOM). Modifying these structures, for instance for applying updates, involves fine surgery for isolating the concerned elements and for applying the intended modifications. As these operations are generally triggered after asynchronous events that may come in response to earlier network requests or a user actions, the programming is complex and error prone. Improving on that situation has been the subject of many previous studies that propose alternative models for helping programming Web UI. Our work constitutes yet another contribution to that problem. It differs from the other solutions by the followings.

- It addresses exclusively the problem of programming the Web UI updates.
- It does not introduce a new programming model and it is fully compatible with traditional JavaScript programming.
- On the client, it only requires a very thin implementation layer whose weight is almost unnoticed in a Web browser.
- It does not impact the rest of the execution, leaving the performances unchanged.

Our proposal consists in introducing a zest of reactive programming used only for denoting the parts of the DOM that need updates. For that, we introduce two new constructs: i) reactive values, called *reactors*, that have the appearance of any regular JavaScript value, and ii) *reactive nodes*, which are DOM nodes that are automatically updated upon reactors changes. Reactors and reactive nodes can be used in pure JavaScript programs but that have been designed to complement other facilities Hop.js. To justify their design and to advocate their benefit, we show how they simplify the programming of classical Web patterns. Let us consider a classical example already detailed in the literature, a timer example, which consists in a simple Web page defined by:

```
var elapsedTime = 0;

function doEverySecond() {
   elapsedTime++;
   document.getElementById( "curTime" )
           .innerHTML = elapsedTime; }

<html>
  <script>setInterval( doEverySecond, 1000 )</script>
  <button onclick="elapsedTime = 0">reset</button>
  <div id="curTime"></div>
</html>
```

Although simple and innocuous at first glance, this program suffers from two major problems. First, the lack of modularity. The function `doEverySecond`, that implements the timer, increments the wall clock *and* updates the UI (via `innterHTML` attribute assignment). Hence, it must be aware of all the elements that needs update. This is problematic as a UI may evolve over time with some elements removed and new elements added. Each evolution of the specification will then impact `doEverySecond` implementation. The second problem we address is the plumbing needed for extracting and modifying the `curTime` element. In the pure JavaScript this involves assigning and looking up unique identifiers (`curTime` identifier). The reactors and reactive nodes we propose solve these two problems.

```
<html> ~{
     const T = hop.reactProxy( { elapsedTime: 0 } );
     setInterval( () => { T.elapsedTime++ }, 1000 );
  }
  <button onclick=~{T.elapsedTime=0}>reset</button>
  <div><react>~{T.elapsedTime}</react></div>
</html>
```

This Hop.js program solves the two problems previously mentioned. It is modular as new reactive elements depending on the `elapsedTime` can be added without modifying existing code. It avoids tedious surgery of the HTML DOM as the `react` node designates the node that need updates and its positioning in the UI.

We have built a first operational prototypes of reactors and reactive nodes. This work will be pursued in 2017. We will complete the implementation in Hop.js by including them in Hop-3.1.0. We will write a scientific paper describing their design and implementation.

5.1.2. Hiphop.js

Modern Web applications are rich in interactions between users and servers. Those interactions are from different nature: search and play music, book train or airplane tickets, query database or use an interactive map. From the programmer point-of-view, those interactions are handled by asynchronous events from multiple sources. Management of those events, which is called orchestration, is done by using event handlers. It is a mechanism that will call a specific function when a specific event raises. This kind of orchestration doesn't scale well since the behavior of the application has to be deduced by the programmer. Synchronous languages like Esterel, which are used in the industrial area, provides syntactic constructs that allow ordering the temporal behavior of the application. Then, reading the program source gives a precise idea of the behavior of the program at runtime.

The HipHop.js contribution is to adapt the reactive constructs of Esterel to the Web. The goal is to design a high-level tool that simplifies the orchestration of Web applications. In the traditional Esterel setting, the reactive program is written in a different source file of the host program. It is compiled independently of the host program. Therefore, the programmer must make explicit bindings between the reactive program and the host program in order to allow both of them to interact. This is inadequate for Web developments. So, HipHop.js adopts a radically different point of view: the reactive program is written in the same source code with the host program and the interaction between the reactive program and the host program is direct, thanks to a JavaScript API which is offered by the compilation output of the reactive program. HipHop.js uses a XML syntax, where each node corresponds to an Esterel instruction. This syntax has pros and cons but we think its advantages dominate. First, it is familiar to all Web developers, which do not have to learn a new syntax. Second, it is overly simple to implement as Hop.js natively supports XML parsing. Third, it gives macros for free as the XML syntax can be mixed with standard JavaScript that can create and return XML objects.

The classical Esterel example of the synchronous community is "ABRO": a program which is waiting for two events in parallel. When both events are raised, the host program is notified (here it pops a window up). At any moment, the reactive program state can be reset, in which case, the reactive program waits again for both events. For the sake of illustration, we show here how to implement ABRO in HipHop.js inside a Web page:

```
<html> ~{
     var abro =
```

```
        <hh.module A B R O>
          <hh.loopeach R>
            <hh.parallel>
                <hh.await A/>
                <hh.await B/>
            </hh.parallel>
            <hh.atom apply=${function() {alert("ABRO")}}/>
          </hh.loopeach>
        </hh.module>

    var m = new hh.ReactiveMachine(abro);
  }
  <button onclick=~{m.inputAndReact("A")}>A</button>
  <button onclick=~{m.inputAndReact("B")}>B</button>
  <button onclick=~{m.inputAndReact("R")}>R</button>
</html>
```

Pushing the buttons “A” and “B” triggers the popup message which contains "ABRO" in the browser page. In spite of its simplicity, the ABRO example is representative of a wide class of real programs. For instance, a program behaviorally similar to ABRO can be used to download a file in several parts of different sources, and merge them when all downloads are completed.

The first HipHop.js version has been released this year. It is available at the following URL http://www-sop.inria.fr/members/Colin.Vidal/hiphop/.

5.1.3. Garbage Collection with non ambiguous roots

Hop uses lot of objets with short time life.

Some Hop programs allocate many temporary objects whose lifetimes are very short. These objects are unefficiently handled by this *Mark&Sweep* garbage collector that Hop currently uses. We expect a speed-up by switching from a *Mark&Sweep* garbage collector to a generational *Stop&Copy* one. *Stop&Copy* collectors demand that all roots of the accessibility graph have to be precisely known (non ambiguous root). We have changed the code generation of the compiler in order to maintain a precise map of the pointers living in the stack.

5.1.4. Event calculus

We have studied functions over streams of events (timed values) and more precisely those which have a temporal causality property: at every instant, current outputs only depends on inputs that have already been received [24]. We have found a clear characterization of causal functions and made some proofs with the Coq system [21].

5.2. Privacy

Participant: Nataliia Bielova.

5.2.1. Hybrid Monitoring of Attacker knowledge

Enforcement of noninterference requires proving that an attacker’s knowledge about the initial state remains the same after observing a program’s public output. We have proposed a hybrid monitoring mechanism which dynamically evaluates the knowledge that is contained in program variables [14]. To get a precise estimate of the knowledge, the monitor statically analyses non-executed branches. We show that our knowledge-based monitor can be combined with existing dynamic monitors for non-interference. A distinguishing feature of such a combination is that the combined monitor is provably more permissive than each mechanism taken separately. We demonstrate this by proposing a knowledge-enhanced version of a no-sensitive-upgrade (NSU) monitor. The monitor and its static analysis have been formalized and proved correct within the Coq proof assistant.

5.3. Security

Participants: Nataliia Bielova, Ilaria Castellani, Tamara Rezk, Dolière Francis Some.

5.3.1. Security for multiparty session calculi

In our previous work, we investigated two security properties for multiparty session calculi: *access control* and *information flow security*. We proposed a type system ensuring both these properties. We also defined a monitored semantics inducing a property that is strictly included between typability and information flow security, which we called *information flow safety*.

The article [5] is an extended version of a previous workshop paper, which introduces refined versions of the safety and security properties examined in that paper and provides two additional results: compositionality of the refined safety property, and the proof that this property is ensured by a simplified version of the type system of [4].

In [18], we argue that the security requirements considered in previous work could be overly restrictive in some cases. In particular, a party is not allowed to communicate any kind of public information after receiving a secret information. The aim of [18] is to overcome this restriction, by proposing a new type discipline for a multiparty session calculus, which classifies messages according to their topics and allows unrestricted sequencing of messages on independent topics.

5.3.2. Security for dynamic and adaptable systems

We have started to study security issues in the context of dynamically evolving communicating systems, namely systems which are able to adapt themselves in reaction to particular events, arising in the system itself or in its environment. When focussing on security, examples of such events are security attacks or changes in security policies.

The paper [11] investigates a simple session calculus in which self-adaptation and security concerns may be jointly addressed. In this calculus, security violations occur when processes attempt to read or write messages of inappropriate security level within a session. Such violations trigger adaptation mechanisms that prevent the violations to propagate their effect in the remainder of the session, while allowing the computation to proceed. More specifically, our calculus is equipped with a monitored semantics based on session types, which activates local and global adaptation mechanisms for reacting respectively to soft and hard security violations. We present type soundness results that ensure that the overall protocol is still correctly executed after the application of these mechanisms.

5.3.3. Information Flow Monitoring

The dynamic aspects of JavaScript make the security analysis of web applications very challenging. Purely static analysis is prohibitively restrictive in practice since it must exclude JavaScript dynamic aspects or over-approximate them. In recent years, several dynamic enforcement mechanisms in the form of information flow monitors have been proposed. In order to better evaluate the currently available information flow monitors trade-offs, our contribution is to rigorously compare them [16]. We compare them with respect to two important dimensions according to the runtime monitor literature: soundness and transparency. We analyse five widely explored information flow monitor techniques: no-sensitive-upgrade, permissive-upgrade, hybrid monitors, secure multi execution, and multiple facets. Furthermore, we formally prove that the generalised belief in the equivalence of two of these approaches, secure multi-execution and multiple facets, is false [17].

5.3.4. Quantitative information flow measures

A number of measures for quantifying information leakage of a program have been proposed. Most of these measures evaluate a program *as a whole* by quantifying how much information can be leaked *on average* by different program outputs. While these measures perfectly fit for static program analyses, they cannot be used by dynamic analyses since they do not specify what information an attacker learns through observing one concrete program output.

In this work, we study the existing definitions of quantitative information flow [15]. Our goal is to find the definition of *dynamic leakage* – it should evaluate how much information an attacker learns when she observes *one program output*. Surprisingly, we find out that none of the existing definitions provide a suitable measure for dynamic leakage. We hence open a new research question in quantitative information flow area: which definition of dynamic leakage is suitable?

5.3.5. Access control and capability systems

Motivated by the problem of understanding the difference between practical access control and capability systems formally, we distill the essence of both in a language-based setting [19]. We first prove that access control systems and (object) capabilities are fundamentally different. We further study capabilities as an enforcement mechanism for confused deputy attacks (CDAs), since CDAs may have been the primary motivation for the invention of capabilities. To do this, we develop the first formal characterization of CDA-freedom in a language-based setting and describe its relation to standard information flow integrity. We show that, perhaps suprisingly, capabilities cannot prevent all CDAs. Next, we stipulate restrictions on programs under which capabilities ensure CDA- freedom and prove that the restrictions are sufficient. To relax those restrictions, we examine provenance semantics as sound CDA-freedom enforcement mechanisms.

6. Partnerships and Cooperations

6.1. National Initiatives

6.1.1. ANR AJACS

The AJACS project (Analyses of JavaScript Applications: Certification & Security) is 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.

6.1.2. 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.

6.1.2.1. Actions marquante

Inria Sophia-Antipolis Actions Marquante is a special funding for 2 postdocs during one year to explore a new research direction. The joint project with DIANA team “User discrimination on the Web: measurement, causation and prevention” has obtained this funding. The goal of this project is to detect when users get discriminated on the Web, what are the technologies used to discriminate users and how we can prevent it without breaking the functionality and sometimes useful personalisation within Web applications.

6.2. European Initiatives

6.2.1. FP7 & H2020 Projects

6.2.1.1. 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.js technology is used by partner academic and SME R&D teams to develop a distributed software platform and applications for assistive robotics.

6.2.2. Collaborations in European Programs, Except FP7 & H2020

6.2.2.1. ICT Cost Action IC1201 BETTY

Program:BETTY

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.

6.2.2.2. ICT Cost Action IC1405 on Reversible Computation

Program: COST

Project acronym: RC

Project title: Reversible computation - extending horizons of computing

Duration: November 2014 - November 2018

Coordinator: Irek Ulidowski, University of Leicester

Other partners: several research groups, belonging to 23 european countries

Abstract: Reversible computation is an emerging paradigm that extends the standard 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 is the first European network of excellence aimed at coordinating research on reversible computation.

6.2.2.3. Bilateral PICS project SuCCeSS

Program: PICS

Project acronym: SuCCeSS

Project title: Security, Adaptability and time in Communication

Duration: June 2016 - June 2019

Coordinator: Cinzia Di Giusto, I3S, Sophia Antipolis

Other partners: I3S, University of Gröningen

Abstract: The project SuCCeSS is a CNRS-funded “Projet coopératif” (PICS 07313), involving 2 French teams in Sophia Antipolis (the MDSC team at the laboratory I3S, acting as coordinator, and the INDES team) and one Dutch team at the University of Gröningen. The project started in June 2016 and is due to end in June 2019. The objective of the project is to study formal models for reliable distributed communication-centric software systems. The project focusses on analysis and validation techniques based on behavioural types, aimed at enforcing various properties (safety, liveness, security) of structured communications.

6.3. International Research Visitors

6.3.1. Visits of International Scientists

6.3.1.1. Internships

Raimil Cruz

Date: 01/05/16 - 30/07/16

Institution: University of Chile

7. Dissemination

7.1. Promoting Scientific Activities

7.1.1. Scientific Events Organisation

7.1.1.1. General Chair, Scientific Chair

- Ilaria Castellani co-organised (with Mohammad Reza Mousavi) the workshop TRENDS 2016, which took place in Québec City in association with the CONCUR conference.

7.1.1.2. Member of the Organizing Committees

- Ilaria Castellani participated in the organisation of the 2nd Summer School of the BETTY project, which took place in Cyprus.

7.1.2. Scientific Events Selection

7.1.2.1. Member of the Conference Program Committees

- Ilaria Castellani served in the programme committee of the workshop EXPRESS/SOS 2016.
- Tamara Rezk served in the programme committee of POST, SEC@SAC, PLAS, IEEE SecDev, and APLAS 2016.

7.1.2.2. Reviewer

The team members have been reviewers for FSTJCS conference, SEC@SAC conference, USENIX Security 2016, POST 2016, NDSS 2016, and CCS 2016.

7.1.3. Journal

7.1.3.1. Member of the Editorial Boards

- Ilaria Castellani is a member of the editorial board of the french journal *Technique et Science Informatiques*.
- Tamara Rezk is a member of the editorial board of the french journal *Interstices*.

7.1.3.2. Reviewer - Reviewing Activities

The team members have been reviewers for the international journals JLAMP (*Journal of of Logical and Algebraic Methods in Programming*), LMCS (*Logical Methods in Computer Science*), International Journal of Information Security (IJIS), EEE TDCS, and ACM TISSEC.

7.1.4. Invited Talks

Manuel Serrano gave a presentation on "Diffuse Web programming" at The Open Source Innovation Spring 2016 (http://open-source-innovation-spring.org/techniques-de-programmation-web-letat-de-lart-date-conf/).

Nataliia Bielova has been invited to give a talk on Price discrimination of Online Airline Tickets at the Collaborative Action on the Protection of Privacy Rights in the Information Society (CAPRIS) project meeting, Sophia Antipolis, France.

7.1.5. Leadership within the Scientific Community

- Ilaria Castellani is the chair of the IFIP TC1 WG 1.8 on Concurrency Theory since May 2014. In this quality, she co-organises every year (together with the WG Secretary Mohammad Reza Mousavi) the annual business meeting of the working group as well as the workshop TRENDS, which is always affiliated with the CONCUR conference. Ilaria Castellani was a member of the european COST Action IC1201 BETTY on Behavioural Types (October 2012-October 2016). She also belonged to the Management Committee of BETTY and was the chair of its working group on security. Ilaria Castellani is a member of the COST Action IC1405 on Reversible Computation (November 2014-November 2018). She is also a deputy Management Committee member of this action.
- Nataliia Bielova has co-organised a Dagstuhl Seminar on Online Privacy and Web Transparency [1] with researchers from Telefonica (Spain), Stony Brook University (USA) and Princeton University (USA).
- Nataliia Bielova is a member of the W3C Tracking Protection Working Group. This group works towards creating a "DoNotTrack" specification. Its goal is to help users express their preferences on third-party tracking and help companies ensure their compliance with the specification.

7.1.6. Scientific Expertise

During 2016, Tamara Rezk has been an international research proposal evaluator for the following research agencies: Ministerio de Ciencia y Tecnología e Innovación Productiva (Foncyt, Argentina), Executive Agency for Higher Education, Research, Development and Innovation Funding (UEFISCDI, Romania).

7.2. Teaching - Supervision - Juries

7.2.1. Teaching

Licence : Vincent Prunet, Algorithms and Data Structures, 8 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 : Nataliia Bielova, Information Flow Security in Web Applications, 15 ETD, University of Pierre et Marie Curie, France

Master : Tamara Rezk, Web Application Security, 28H ETD, University of Nice Sophia Antipolis, France

Master : Tamara Rezk, Proofs of Cryptography, 28H ETD, University of Nice Sophia Antipolis, France

Master : Tamara Rezk, Information Flow Security in Web Applications, 15 ETD, University of Pierre et Marie Curie, France

7.2.2. Supervision

PhD in progress : **Colin Vidal**, *Programmation Web réactive*, University of Nice, 1/07/2015, **Manuel Serrano** and **Gérard Berry**.

PhD in progress : **Dolière Francis Some**, *Web Tracking Prevention*, University of Nice, 1/11/2015, **Nataliia Bielova** and **Tamara Rezk**.

PFE Master in progress: Leila Kuntar, Analysis of web tracking technologies that use 1x1 pixel image, University of Nice, Nataliia Bielova

7.2.3. Juries

- Nataliia Bielova was an examiner of the PhD thesis of Willem De Groef, KU Leuven, Belgium.
- Ilaria Castellani was a member of the "Comité de sélection " for a position of Maître de conférences at the University of Paris Diderot (Paris 7), IRIF Laboratory.

[1] https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=17162

- Ilaria Castellani was a reviewer of the PhD thesis of Saverio Giallorenzo, University of Bologna. She took part in the jury of this PhD thesis, as well as of ten other PhD theses defended in the same round.
- Ilaria Castellani was an examiner for the PhD thesis of Aurélien Deharbe, Université Pierre et Marie Curie (Paris 6).
- Tamara Rezk was an examiner of the PFE juries for Security and Privacy and Mobile Cloud and IoT at Univesity of Nice Sophia Antipolis.
- Manuel Serrano was an examiner of the PhD thesis of Rabah Laouadi, Université de Montpellier.
- Manuel Serrano was an examiner of the PhD thesis of Julien Pagès, Université de Montpellier.

7.3. Popularization

- Web users are continuously tracked as they browse the Web. One of the techniques for tracking is device fingerprinting that distinguishes users based on their Web browser and operating system properties. Together with Inria Celtique team, we have proposed solutions to detect and prevent device fingerprinting via runtime monitoring of JavaScript programs. We have published an article in a general public ERCIM News magazine [2] about prevention of device fingerprinting via program monitoring [10].
- We have implemented the *WebStats* website [3]. This website collects, on a monthly basis, a number of JavaScript and security statistics about top 10 000 webpages: the usage of popular JavaScript libraries; the usage of different language constructs in these libraries; use of Content Security Policies and secure cookies, etc. The *WebStats* website can be used by programmers (to understand which JavaScript libraries are more popular), researchers in programming languages (when designing a subset of JavaScript, to safely exclude the language constructs that are rarely used according to *WebStats*), and researchers in privacy (to analyse which tracking libraries are the most prevalent).

7.4. Transfer

7.4.1. WebRobotics

The WebRobotics initiative aims at developing collaborations with partner academic and industry teams to jointly prototype and experiment end user applications involving assistive robots and sensor devices (depending on the size and number of the embedded components, applications may be either classified as robotic or IoT ones). Each WebRobotics project is structured around partner medical institutions that provide key requirements to specifications and use the actual prototype throughout their daily activity. WebRobotics Applications all use Hop.js as their core framework, natively supporting web protocols for communication and distribution of tasks, and any web enabled device such as a smartphone or tablet to drive the robots and applications. In 2016, the initiative accounted to two full time engineers until the completion of the project, mid year.

The Top Three Benefits of WebRobotics:

- WebRobotics focuses on key societal issues, developing real applications for demanding users.
- Application developers and users feedback to Hop.js framework developers, helping identify and prioritize key requirements.
- The WebRobotics application portfolio fosters the dissemination and transfer of the Hop.js technology to the Industry.

[2] http://ercim-news.ercim.eu/en106/special/using-javascript-monitoring-to-prevent-device-fingerprinting
[3] http://webstats.inria.fr/

The WebRobotics initiative now encompasses several prototypes in use by medical foundations and hospitals.

- RAPP. The WebRobotics project is part of the RAPP FP7 european project, to be completed in December 2016, where Hop.js technology is used by several academic and SME R&D teams to develop a distributed software platform and applications for assistive robotics. Two prototypes have been developed, the first one is a personal coach robot (a Nao humanoid robot embedding Hop.js 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). Both prototypes are being evaluated by partner medical institutions.
- Hopcare. Indes collaborates 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.js technology. An expert engineer is dedicated to this project (grant from UCN@Sophia Labex, until April 2016).
  - ICP (Institut Claude Pompidou Hospital, in Nice) is now using the Alzheimer diagnosis tool developed using Hop.js. User Data generated from Inria/Stars sensors and image analysis software are collected by a Hop.js server and processed before being delivered to the Physician's web tablet, as an editable web report, or paper ready PDF reports.
  - The activity monitoring application enables real-time monitoring of various events generated by hardware/software monitoring tools (such as the video monitoring applications from Inria/Stars) as well as user defined events. Hop.js is the common framework for the whole application (communications with remote information servers, processing of input data, database management, user authentication and authorization, custom views for web clients). The application will soon be deployed at the Nice Valrose EHPAD (a specialized institution for elderly who need medical care), where Inria runs an experimentation lab.
  - A third application has been developed to enable the configuration and use of Inria/Stars video analysis tools through a web interface. The application is used by researchers to tune their data processing algorithms.

7.4.2. Hop.js for IoT

As more and more software developers come to IoT, teams are facing critical challenges due to the inherent complexity of multi-platform distributed development, leading to team building issues, long and costly development cycles to deliver products with the highest quality, usability, and security standards

The Hop.js software suite enables agile software teams to build flexible, robust and secure end-to-end IoT applications with a single language and a consistent set of API and built-in software components.

Building on the Hop.js technology and the successful WebRobotics experiments, a startup project has been launched in 2016 as a spin-off of the Indes team, with the support of Inria DGD-T, funding two engineers from July 2016.

The team has initiated partnerships with IoT hardware vendors, adapted Hop.js to highly constrained execution environments on microcontrollers, and participated to a number of public and business events to promote the solution and meet future customers. The startup company is expected to launch in 2017.

8. 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] 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

[5] S. CAPECCHI, I. CASTELLANI, M. DEZANI-CIANCAGLINI. *Information Flow Safety in Multiparty Sessions*, in "Mathematical Structures in Computer Science", 2015, vol. 26, n^{o} 8, 43 p. [*DOI :* 10.1017/S0960129514000619], https://hal.inria.fr/hal-01237236

[6] 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

[7] 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^{o} 8, pp. 53–59 [*DOI :* 10.1145/2240236.2240253], http://cacm.acm.org/magazines/2012/8/153796-multitier-programming-in-hop/abstract

[8] 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

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

Publications of the year

Articles in International Peer-Reviewed Journals

[10] N. BIELOVA, F. BESSON, T. JENSEN. *Using JavaScript Monitoring to Prevent Device Fingerprinting*, in "ERCIM News", July 2016, https://hal.inria.fr/hal-01353997

[11] I. CASTELLANI, M. DEZANI-CIANCAGLINI, J. A. PEREZ. *Self-adaptation and secure information flow in multiparty communications*, in "Formal Aspects of Computing", 2016, vol. 28, n^{o} 4, 28 p. [*DOI :* 10.1007/S00165-016-0381-3], https://hal.inria.fr/hal-01354906

[12] Z. LUO, J. FRAGOSO SANTOS, A. ALMEIDA MATOS, T. REZK. *Mashic compiler: Mashup sandboxing based on inter-frame communication*, in "Journal of Computer Security", 2016 [*DOI :* 10.3233/JCS-160542], https://hal.inria.fr/hal-01353966

[13] S. E. REPPOU, E. G. TSARDOULIAS, A. M. KINTSAKIS, A. L. SYMEONIDIS, P. A. MITKAS, F. E. PSOMOPOULOS, G. T. KARAGIANNIS, C. ZELIENSKI, V. PRUNET, J.-P. MERLET, M. ITURBURU, A. GKIOKAS. *RAPP: A Robotic-Oriented Ecosystem for Delivering Smart User Empowering Applications for Older People*, in "International Journal of Social Robotics", June 2016 [*DOI :* 10.1007/S12369-016-0361-Z], https://hal.inria.fr/hal-01336250

International Conferences with Proceedings

[14] F. BESSON, N. BIELOVA, T. JENSEN. *Hybrid Monitoring of Attacker Knowledge*, in "29th IEEE Computer Security Foundations Symposium", Lisboa, Portugal, 2016, https://hal.inria.fr/hal-01310572

[15] N. BIELOVA. *Dynamic leakage - a need for a new quantitative information flow measure*, in "Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security", Vienna, Austria, October 2016, pp. 83-88 [*DOI :* 10.1145/2993600.2993607], https://hal.inria.fr/hal-01409706

[16] N. BIELOVA, T. REZK. *A Taxonomy of Information Flow Monitors*, in "International Conference on Principles of Security and Trust (POST 2016)", Eindhoven, Netherlands, F. PIESSENS, L. VIGANÒ (editors), LNCS - Lecture Notes in Computer Science, Springer, April 2016, vol. 9635, pp. 46–67 [*DOI :* 10.1007/978-3-662-49635-0_3], https://hal.inria.fr/hal-01348188

[17] N. BIELOVA, T. REZK. *Spot the Difference: Secure Multi-Execution and Multiple Facets*, in "European Symposium on Research in Computer Security (ESORICS)", Heraklion, Greece, September 2016, https://hal.inria.fr/hal-01348192

[18] I. CASTELLANI, M. DEZANI-CIANCAGLINI, U. DE'LIGUORO. *Secure Multiparty Sessions with Topics*, in "PLACES 2016", Eindhoven, Netherlands, Proceedings of the 9th workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software, PLACES 2016, Eindhoven, The Netherlands, 8th April 2016., Dominic A. Orchard and Nobuko Yoshida, April 2016, vol. 211, 12 p. [*DOI :* 10.4204/EPTCS.211.1], https://hal.inria.fr/hal-01354905

[19] V. RAJANI, D. GARG, T. REZK. *On Access Control, Capabilities, Their Equivalence, and Confused Deputy Attacks*, in "Computer Security Foundations", Lisbon, Portugal, June 2016 [*DOI :* 10.1109/CSF.2016.18], https://hal.inria.fr/hal-01353963

[20] M. SERRANO, V. PRUNET. *A Glimpse of Hopjs*, in "International Conference on Functional Programming (ICFP)", Nara, Japan, ACM, September 2016, 12 p. [*DOI :* 10.1145/2951913.2951916], https://hal.inria.fr/hal-01350936

National Conferences with Proceedings

[21] B. P. SERPETTE, D. JANIN. *Causalité dans les calculs d'événements*, in "JFLA 2017 - Vingt-huitième Journées Francophones des Langages Applicatifs", Gourette, France, January 2017, https://hal.inria.fr/hal-01403369

Scientific Books (or Scientific Book chapters)

[22] M. SERRANO. *The Computer Scientist Nightmare: My Favorite Bug*, in "A List of Successes That Can Change the World : Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday", Lecture Notes on Computer Science, Springer, April 2016, vol. 9600, pp. 356-366, https://hal.archives-ouvertes.fr/hal-01340384

Books or Proceedings Editing

[23] M. SERRANO, J. HAGE (editors). *Trends in Functional Programming (TFP 2015): Revised Selected Papers*, Lecture Notes on Computer Science, Springer Verlag, Sophia Antipolis, France, February 2016, vol. 9547, 156 p. [*DOI :* 10.1007/978-3-319-39110-6], https://hal.inria.fr/hal-01354237

Research Reports

[24] D. JANIN, B. P. SERPETTE. *Timed Denotational Semantics for Causal Functions over Timed Streams*, LaBRI - Laboratoire Bordelais de Recherche en Informatique, November 2016, https://hal.archives-ouvertes.fr/hal-01402209

[25] B. P. SERPETTE. *Logical semantics of Esterel with unconstrained local signals*, Inria Sophia Antipolis - Méditerranée, August 2016, n^{o} RR-8942, https://hal.archives-ouvertes.fr/hal-01351005

[26] B. P. SERPETTE. *Using counters for absence prediction in Esterel*, Inria Sophia Antipolis - Méditerranée, July 2016, n^{o} RR-8941, 18 p. , https://hal.inria.fr/hal-01226760