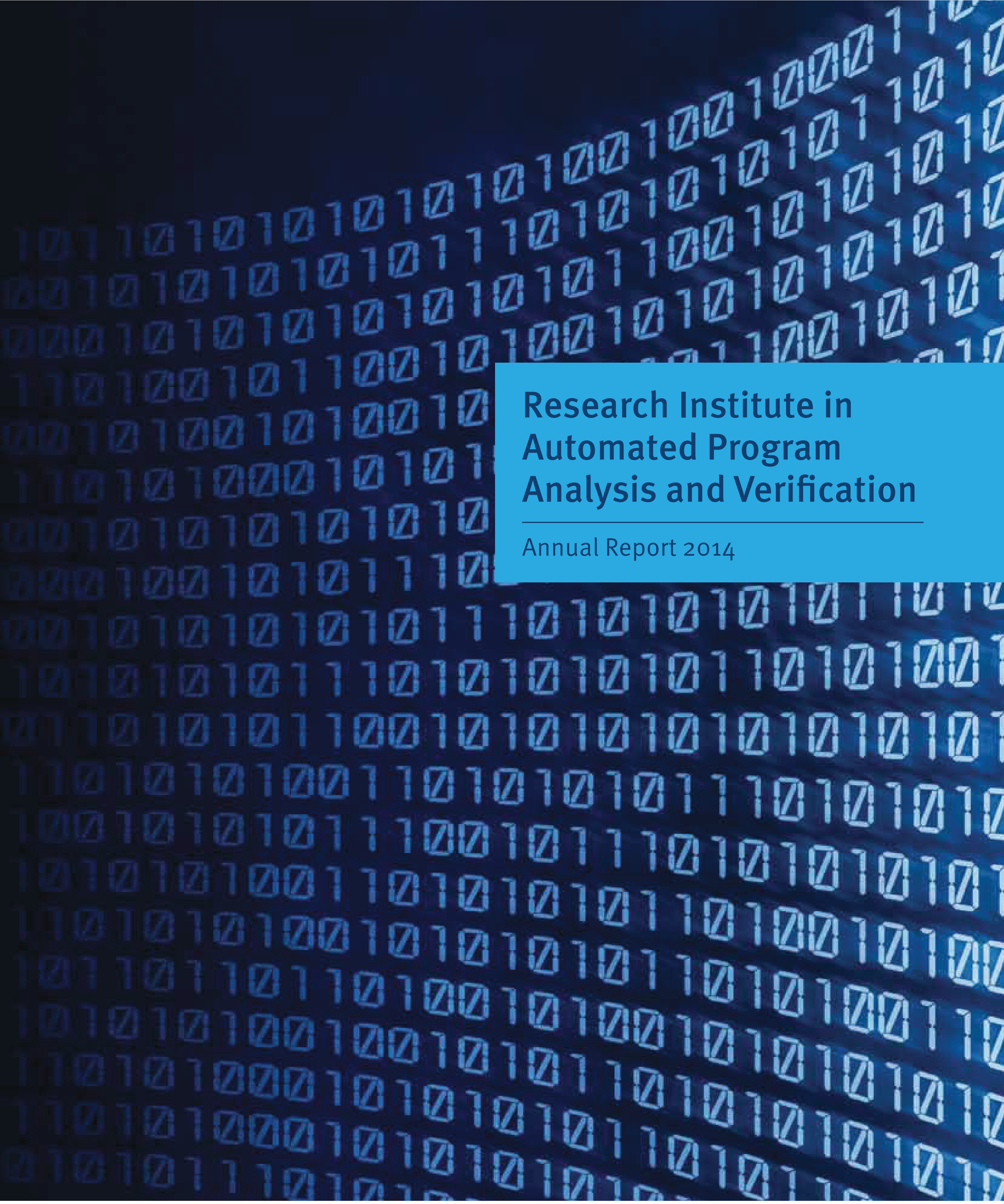


Research Institute in Automated Program Analysis and Verification

Annual Report 2014

GCHQ

Department for Business Innovation & Skills

EPSRC Engineering and Physical Sciences Research Council

Advisory board

Participating universities

Imperial College London

The University of Edinburgh

MANCHESTER 1824 The University of Manchester

UCL

University of Kent

Queen Mary University of London

Foreword

The Research Institute in Automated Program Analysis and Verification is the UK’s second Academic Research Institute in cyber security. It has been running for almost a year and comprises of six projects based at the University of Edinburgh, Imperial College London, University of Kent, The University of Manchester, Queen Mary University of London and University College London. The common theme is to advance UK research in automated program analysis and verification, in particular with its application to cyber security. We will improve the security of our software systems by providing greater understanding, proving correctness and identifying potential weaknesses of our software.

In this relatively short period, our six projects have made excellent progress as evidenced by our invited keynotes and our publications in high-impact conferences and journals. We have already had substantial interaction with government (primarily GCHQ), professional bodies such as the Royal Society, our international advisory board, industry and universities via our website (www.verificationinstitute.org), face-to-face meetings, reports, presentations and workshops. One of our highlights was co-organising the third workshop on *Formal Methods and Tools for Security* at Microsoft Research Cambridge (www.fmats.org), which brought together leading experts for two days of stimulating talks and discussions.

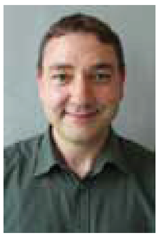
We hope that you will find this annual report of interest.

Professor Philippa Gardner
Director

App Guarden: Resilient Application Stores

Aims to improve resilience of application stores by producing methods to automatically analyse and sign apps for safety properties.

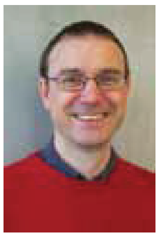
Principal Investigator


David Aspinall

Co-Investigators

Andrew Gordon

Don Sannella


Ian Stark

Charles Sutton

Björn Franke

Industrial partners: Google New York, RIM, McAfee, Kotican, Metaforic
Academic partners: LMU Munich, UCM Madrid, Birmingham University, Glasgow Caledonian

Application stores are set to become the dominant model for software distribution. After only four years, they had become incredibly successful: in 2012, Apple's App Store and Google's play store each topped 25 billion app downloads. App stores not only offer apps and media content, they also have near total control on phones and tablets that connect to them. Hundreds of millions of people place their trust in app store and device security every day. Unfortunately, this trust is sometimes misplaced and is starting to be eroded.

App stores of the future, and the devices they control, must be better defended and resilient under attack. Users and data owners need justifiable confidence that apps will behave well and will not cause damage, whether by accident through bugs, or by intention through malicious design. Security should be ever present but unobtrusive, not impacting performance or causing crashes, not forever downloading patches, not demanding complex decisions, and not in the hands of just one party.

Our research will examine a number of improvements to app stores and mobile device operating systems which will take us closer to future generation, secure app stores.

For example, we will design algorithms that will automatically analyse apps to ensure they are safe. At the moment, this has to be done manually by malware analysts in expensive, time-consuming and sometimes unreliable ways. Another improvement is to add 'digital evidence' to apps. Digital evidence can guarantee that an app is safe and it can be checked automatically, even on a phone. Evidence establishes that the code is safe, whereas the current state-of-the-art in industry is code signing, which at best only says where the code has come from. Finally, we want to find natural, user-friendly security policies: rather than the user examining a long list of complicated permissions as currently happens in Android, we want to have a set of sensible policies for different types of app. Under the bonnet the controls will actually be more precise than at present: with our solution, a game, for example, would not be allowed to access anywhere on the Internet, just the few places that it really needs to go; a text-messaging app might only be allowed to send messages to contacts from a users address book, not unknown numbers that might be premium-rate.

Key milestones achieved

- Proof of concept of digital evidence with a reachability checking tool and simple app policy language.
- Malware classifier trained with McAfee data (accuracy 95%, TPR 96%, FPR 5%), together with compact explanations of abnormal behaviours for each category of app.
- Experimental study using the EviCheck tool, checking a set of anti-malware probes (non-reachability policies) on 300 popular apps from the Google play store and 300 malware samples provided by McAfee.
- Using machine learning methods: defining malware by its suspicious behaviours against normal behaviours of its comparison groups.
- Invented an approach for the automatic inference of lightweight policies for EviCheck.
- Work on a method to produce *Compact Explanations of Why Malware is Bad*, which uses machine-learning driven malware characterisation by difference in behaviour against normal behaviours in comparison groups.

Other activities

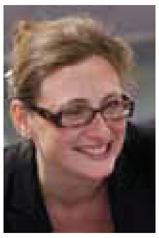
- Started on related side-project *App Security Knowledgebase* which is building a research platform for hosting large numbers of Android applications and the results of running analysis tools on them.
- A connected PhD project started to work on a language for expressing device-level policies using SecPAL; paper presented at Doctoral Symposium in ESSOS 2014.
- Stimulating visits from David Barrera (Carleton) and Lorenzo Cavallaro (Royal Holloway), supported by GCHQ Small Grant. Barrera is sharing his repository of approx 50k Android apps and Cavallaro plans to collaborate in future, with dynamic analysis techniques of his CopperDroid platform complementing static analysis techniques from App Guarden.
- Visit from Ulfar Erlingsson, Head of Security Research at Google. We learned about Google's approach to handling malware.
- Attended and gave talks at FMATS3 (Cambridge), NIMBUS workshop (Royal Holloway), CryptoForma (York, UK).
- Short visit from Eric Bodden, Head of Secure Software Engineering Group at EC SPRIDE, Darmstadt, Germany. Held discussions on App Guarden project and their FlowDroid tool.
- Research Masters thesis completed by visiting MSc student, who built the backend for the *App Security Knowledgebase* and seeded analysis of an initial set of apps and tools.

Imperial College London

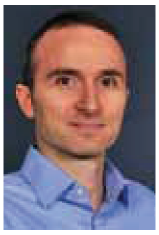
Certified Verification of Client-Side Web Programs

Aims to provide a mechanised specification of the JavaScript standard on which to develop theories and tools for proving correctness, safety and security properties of JavaScript programs.

Principal Investigator


Philippa Gardner

Co-Investigator


Sergio Maffeis

Industrial partners: Mozilla Foundation, Google California
Academic partners: INRIA Rennes, KU Leuven

The web is evolving at enormous speed from a collection of mainly static web pages to the current huge dynamic ecosystem where the boundary between web pages and software application has become indistinct (e.g. Google Maps). This effect is so pronounced that industry is beginning to view the web as an operating system: e.g. Google's Chrome OS and Firefox OS. This quick transformation has come at a price. We are stuck with dynamic languages developed for the early web. These languages are unsuited to the development of sophisticated web applications, resulting in modern applications being either overly conservative or needlessly unreliable and insecure. The web will only be trustworthy if the programs that are used to access it are robust, reliable and secure.

JavaScript is the most widely used language for the web. All programs written for the browser are either written directly in JavaScript or in other languages (e.g. Google's Dart) which compile to JavaScript. JavaScript is currently the assembly language of client-side web programming. It is the only language supported natively by all major web browsers, and this fundamental role seems unlikely to change. JavaScript was initially created for small web-programming tasks, which benefited from the flexibility of the language and tight browser integration. Nowadays, the modern demands placed on JavaScript are vast. Although flexibility and browser integration are still key advantages, the inherent dynamic nature of the language leads to buggy programs that cannot be trusted. JavaScript offers little support for modularity, has no reliable IDE support, and has large numbers of tricky corner cases masquerading as simple, intuitive programs. Using it to write secure programs is extremely difficult.

Verification has much to offer JavaScript: a simple description of program behaviour; the safe composition of programs; a clear understanding of conceptual module boundaries; and the ability to verify security contracts. E.g. we should be able to assert that a particular web application will maintain the structure of a web page and will not leak secret data, or that a browser extension will only perform permitted file system operations. There has been some work on formally analysing JavaScript programs that has been helpful for discovering bugs and for describing specific safety problems. However, none provide general-purpose verification analyses, most do not work with the full language and, of those that prove soundness, all do so with respect to their abstract models rather than the ECMA semantics or an actual concrete implementation.

Our project will provide a general-purpose verification tool for proving correctness, safety and security properties of JavaScript programs, based on our operational semantics and program logic. We aim to enable and to provoke a profound change in how people approach verification research and design programs for the web. We advocate the development of certifying verification tools, supported by Coq (a mechanical proof assistant for checking and discovering formal proofs) and with a strong empirical link to industrial languages. These verification tools will provide an essential foundation to underpin the design of specialised development tools for the engineers that build the web thereby bringing verification to mainstream web development. There has been much success in developing automatic verification tools for C language programs based on separation logic. We will build on this experience to develop an automatic verification tool for JavaScript. However, our approach is different. Our tool will use unsound heuristics to automatically generate proofs and then certify that the proofs are correct by checking them in Coq. This way we obtain the maximum trust in the program specifications generated by our verification tool.

Our ambitious goal is to ensure that the software we use to communicate with our banks is at least as reliable as the software as our banks use to communicate with each other.

Objectives

1. JSCert, a formal mechanised specification (in Coq) of the English standard specification, ECMAScript 5 validated by JSRef;
2. JSRef, a JavaScript reference interpreter, automatically generated from JSCert and hence Coq- certified, which will be tested to industrial standards;
3. JSVerify, a certifying general-purpose verification tool for JavaScript, whose automatically generated proofs are checked using JSCert;
4. a plugin architecture for separation-logic verification tools such as JSVerify and Verifast to enable e.g. DOM, CSS, JQuery and Node.js library plugins;
5. the certified compilation of secure web languages such as secure JavaScript subsets and Miller's Secure ECMAScript (SES) into JSCert and JSVerify respectively, to prove that they are indeed secure;
6. usable tools based on JSVerify, targeted at specific needs of the JavaScript developer community: e.g. understanding which JavaScript code touches which elements of a page, security analysis of browser extensions, and automatically generating fail-early test cases to assist in the investigation of complex bugs.

Key milestones achieved

- JSCert, a mechanised specification of JavaScript, written in the Coq proof assistant, which closely follows the ECMAScript 5 English standard: see **http://jscert.org**
- JSRef, a reference interpreter for the core JavaScript language (chapters 1–14 of the ECMA standard) in OCaml, which has been tested with the Test 262 test suite. We are now working with an intern (Conrad Watt) on the standard libraries (chapter 15 of the ECMA standard).
- A simple analysis using the OCaml 'bisect' tool, of how much of the ECMA standard is actually tested by the Test 262 test suite.
- A mechanically checked proof (in Coq) which connects the behaviour of JSRef with the specification JSCert. This means that tests of JSRef validate the JSCert interpretation of ECMAScript 5, and that the JSCert closeness to the standard lends confidence in the behaviour of JSRef.
- Extension of our JavaScript program logic (POPL 2012) to deal with higher-order functions.
- Initial work on program logic for Secure ECMAScript (SES).
- Invited talks and department seminars on JSCert at JSTools'13 (ECOOP), PiP'14 (POPL), Mozilla (2013) and Google California (2013), University of Kent, University College London, University of York, University of Verona, FMATS3 and Dagstuhl.
- Participated in sotu.js in San Jose. This was a one-day event, plus informal networking with leaders from industry and academia in the JavaScript space.
- Participated in Dagstuhl 14271 meeting on *Scripting Languages and Frameworks: Analysis and Verification* (June 30–July 4). This was a one-week invited residential workshop for experts on scripting languages (such as JavaScript).
- Hosting ENS Cachan internship, working on *Reduction-closed invariance proof for JSCert* and GCHQ sponsored intern (Conrad Watt) on extending the coverage of JSRef.

Papers

– *A Trusted Mechanised JavaScript Specification,* M. Bodin, A. Charguéraud, D. Filaretti, P. Gardner, S. Maffeis, D. Naudziuniene, A. Schmitt, and G. Smith, POPL 2014.

In preparation:
– *Higher order JavaScript*
– *Journal paper on program logic*
– *Journal paper on JSCert and JSRef*
– *SES*

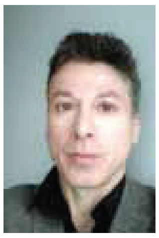
Other activities

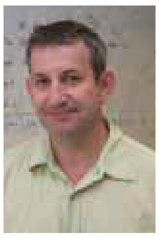
- Partners in INRIA are working on using JSCert as a base for an information flow analysis for secure JavaScript.
- We are partners on a newly funded INRIA's project that uses, extends and complements JSCert.
- Sergio Maffeis supervised BEng projects by Charlie Hothersall-Thomas (prize-winning): *BrowserAudit: A web application that tests the security of browser implementations* (released at: www.browseraudit.com) and by Tomos Jenkins: *The browser security barometer* (pre-release: www.vdetect.uk).
- Thomas Wood, 1st year PhD student of Philippa Gardner, is working on automatically generating high quality tests from mechanised formal specifications. For example, to supplement the Test262 suite, which tests whether the JavaScript engines used in web browsers actually conform to the ECMA standard.
- Sergio Maffeis selected as Co-Chair of Security Track at ACM Symposium On Applied Computing 2015.

Compositional Security Analysis for Binaries

Aims to develop a framework and tools for scalable security analysis for binaries.

Principal Investigators


Pasquale Malacaria


Andy King

Byron Cook

Co-Investigator

Michael Tautschnig

Binaries are routinely inspected by the intelligence community, military organisations and security engineers in their search for vulnerabilities. Binaries are often huge and therefore verification and program analysis techniques should be scalable. This project will pioneer compositional analyses for binary code. This will result in analyses that are both modular and scalable. The scientific challenge in compositional reasoning is how to separate intricate interactions, avoid expensive operations such as quantifier elimination, and derive procedure summaries that are compact. The project team will develop foundational techniques for the compositional analysis of binaries, testing their viability with a running case study: the data sanitisation problem. Confidential data is sanitised when its memory is zeroed before it is deallocated, preventing an attacker retrieving the sensitive information. Data sanitisation is scientifically fascinating because of the need to track how secrets are passed from one procedure to another and, in addition, how secrets are embedded into compound data-structures. The problem is exacerbated by up-casting and down-casting, and the need to track the size of a data object to ensure, for example, that all the elements of a buffer are properly zeroed.

Key milestones achieved

- Developed framework for security analysis of binaries based on self-composition.
- Developed Intermediate Representation (IR) for translating X86 binaries to C code.
- Implemented translator from binaries to CBMC via IR.
- Tested the implementation on sanitized and non-sanitized binaries.
  - CBMC winning 2014 TACAS verification competition.
- Improvements to translation of X86 binaries to C code.
- Translation of ARM binaries to intermediate representation.
- Developed IC3 approach for security analysis.
- Review of open-source applications for potential information leaks due to lack of sanitisation.
- Success in GCHQ small grant applications.
- CBMC based analysis of Heartbleed bug.
- Developed new generic infrastructure for control-flow recovery and instruction decoding.
- Developed new approach for bottom-up (compositional) type recovery.
- Further refinement of IC3 based method for security analysis.
- Verification of openSSL for security leaks.

Papers

– Ed Robbins, Andy King and Jacob M. Howe. *(2013) Theory Propagation and Rational-Trees*. Principles and Practice of Declarative Programming. ACM Press, pp. 193–204.

– Klaus Dräger, Vojtech Forejt, Marta Kwiatkowska, David Parker and Mateusz Ujma. *Permissive Controller Synthesis for Probabilistic Systems*. TACAS 2014.

– Quoc-Sang Phan and Pasquale Malacaria. *Abstract Model Counting: a novel approach for Quantification of Information Leaks,* in Proceedings ACM ASIACCS 2014 Kyoto.

– Robbins, Ed and Howe, Jacob M and King, Andy. *(2014) Theory Propagation and Reification*, Science of Computer Programming, to appear.

– Quoc-Sang Phan, Pasquale Malacaria, Corina Pasareanu and Marcelo D'Amorim. *Quantifying Information Leaks using Reliability Analysis*, in Proc Spin 2014.

Other activities

- On-going development of CBMC and IR.
- On-going development of IC3 based algorithmic approach.
- Set up of GCHQ supported infrastructure for large-scale benchmarking.

Program Verification Techniques for Understanding Security Properties of Software

Aims to develop automatic program verification methods (drawing on static and dynamic techniques) that help security engineers to understand software that they have not written themselves.

Principal Investigator

Brad Karp

Co-Investigators

Mark Handley

Byron Cook

Juan Navarro Perez

Industrial partners: Google USA, Microsoft (Trustworthy Computing Group, UK), Microsoft Research (Cambridge, UK)

This project aims to develop automatic program verification methods that help security engineers to understand software that they have not written themselves, and enforce security policies for such software. A further aim is to provide security engineers with policy enforcement primitives they can use to write software that robustly preserves the user's privacy. The engineer will be able to make sophisticated queries about resource requirements and temporal behaviour of code, such as about memory safety, privileges, or information flow. Our methods will even support synthesis of behavioural properties for the engineer: rather than make a closed-world assumption, where the complete program and physical computing device are known, our tools will discover logical descriptions of execution environments (preconditions, protocols, invariants, etc.) that pinpoint the assumptions necessary for code safety or those that trigger violations. Such tools would aid engineers by, for example, advising where to concentrate effort when looking for critical security breaches. They would also suggest where to place effort in hardening an application. Finally, by using strong analysis techniques based on verification, guarantees of security properties could be obtained, as well as flaws discovered.

Towards realising this vision we have assembled a team whose experience ranges from program verification research on logics and algorithms to systems security research involving new operating system primitives and software structuring principles that achieve robust security goals.

Key milestones achieved

- Designed the G8 system for enforcing security policies on JavaScript applications built for Node.js. G8's policies work by enforcing information flow control. Example security policies include prevention of leakage of sensitive data from applications and prevention of remote script injection by adversaries. G8 works for real-world 'legacy' Node.js applications.
- Produced working initial interpreted prototype of G8 system in Google's V8 JavaScript engine for Node.js; presented design and implementation at RI2 meeting and FMATS3.
- Demonstrated G8 detecting and blocking exploit of a remote command injection vulnerability in the widely used EtherPad collaborative document editor application.
- Extended G8 to allow the merging of policy labels, essential for concurrent enforcement of multiple policies.
- Identified directory traversal vulnerability in a Node.js application; implemented policy in G8 to prevent exploit of this class of vulnerability.
- Designed a performance-enhanced approach to implementing G8 based on source-to-source JavaScript compilation with the Closure compiler, to leverage the V8 engine's just-in-time compiler.
- Designed COWL (Confinement with Web Origin Labels), a label-based mandatory access control system for web browsers (work in collaboration with Google, Mozilla, and Stanford). COWL provides strong confinement of untrusted JavaScript in web applications. As a result, it provides strong protection against leaks of sensitive information from web applications and enables the creation of 'mashup' web applications that are today impossible to implement securely.
- Built full prototype implementations of COWL for the Firefox and Chromium web browsers. Experiments show that COWL offers strong privacy at very low cost: the overhead introduced by COWL is imperceptible by browser users.

Papers

– Yang, E., Stefan, D., Mitchell, J., Mazières, D., Marchenko, P., and Karp, B., *Toward Principled Browser Security,* in the Proceedings of the Fourteenth Workshop on Hot Topics in Operating Systems (HotOS 2013).

– Stefan, D., Yang, E., Marchenko, P., Russo, A., Herman, D., Karp, B., and Mazières, D., *Protecting Users by Confining JavaScript with COWL,* to appear in the Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2014).

Other activities

- Productive collaboration visits by Stefan and Yang of Stanford University to University College London (UCL) while working on COWL.
- Productive collaboration visit by Karp of UCL to Stanford University while working on COWL.
- Ongoing work on developing efficient sandboxing techniques for server applications based on Software-Based Fault Isolation (SFI). This strand of work incorporates program analysis as an *optimisation* to apply *opportunistically* (in this case, to eliminate SFI guard instructions when analysis succeeds and suggests they are unnecessary).
- Ongoing work on implementing the performance-enhanced G8 design based on source-to-source compilation in the Closure JavaScript compiler.
- Ancillary news from our other systems research: paper (with Salameh, Zhushi, Handley, Jamieson, and Karp, all of UCL), *HACK: Hierarchical ACKnowledgments for Efficient Wireless Medium Utilization*, received the Best Paper Award at USENIX ATC 2014.
- Brad Karp selected as Program Co-Chair of ACM SIGCOMM 2015 conference, to be held in London in August 2015.

The University of Manchester

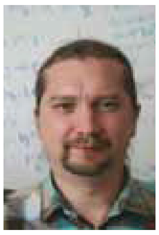
REVES: REasoning in VErification and Security

Aims to enhance first-order theorem provers to use them in program analysis (Vampire) and to develop methods for verifying access policies in web services using such provers.

Principal Investigator

Andrei Voronkov

Co-Investigator


Konstantin Korovin

Academic partners: TU Vienna, Chalmers University of Technology

This project focuses on advancing reasoning-based verification and security analysis of software and web services. In our everyday life we rely on security of software and web services e.g. when using digital banking or social networks and therefore the problem we are addressing is both challenging and important. This problem is highly non-trivial and one of the major challenges comes from the enormous complexity and growing size of the software used in security-critical applications. Typically such software contains from hundreds of thousands to millions lines of code written by different developers using different platforms and requirements. How we can ensure that these complex software systems are functioning correctly and do not have security vulnerabilities? Our approach is to develop fully automatic methods and tools for verification and security analysis based on rigorous mathematical foundations. These methods are based on formalisation of the verification problem in formal logic and applying automated theorem proving to prove that the security properties are satisfied, or otherwise find security vulnerabilities if such a proof fails. Over 50 years of research in automated theorem proving resulted in deep theoretical results and powerful tools based on these results. Our group is world-leading in this area, our theorem proving systems (Vampire and iProver) have been winning almost all major divisions in the world cup in first-order theorem proving (CASC) over the last years. However, program verification and security analysis requires further considerable advances in both theorem proving and formalisation which we address in this project.

The project consists of three major parts:

A. Automatic generation of program properties using symbol elimination and interpolation.

B. Application of theorem provers in verification of real-life large-scale web services.

C. Efficient reasoning with quantifiers and theories with applications in verifying program properties.

Part A: continues the line of research in algorithms for an automatic generation of program properties we started in 2009. Generation of such properties is very important for analysing very large programs, including checking their security-related features.

Part B: aims to design a practical low-cost methodology for verification or access policies for large-scale web services, demonstration of viability of this methodology by verifying a real-life web service, and supporting this methodology by tools based on theorem provers and model finders.

Part C: is rooted in our understanding that efficient reasoning with both quantifiers and theories is crucial for applications of theorem provers in verification and program analysis and will be central in automated reasoning research for the next decade or even longer. It aims at the design and implementation of efficient algorithms for automated reasoning when both quantifiers and theories are used.

Key milestones achieved

- With our colleagues, we have developed a method for efficient first-order theorem proving about collections (arrays, sets, maps). The method was implemented in our theorem prover, Vampire.
- We undertook major updates in Vampire, removing about 25% of the existing code. The updates will make the system more easily maintainable. Most of the code removed was related to splitting features, which makes new developments very hard, yet are not necessary since our new AVATAR architecture makes old ways of splitting essentially unnecessary.
- Together with collaborators from Microsoft, Carnegie Mellon Software Engineering Institute and Tel Aviv University we have been developing new methods for EPR-based interpolation. The results are mainly theoretical; developing a more practical algorithm for interpolation is future work.
- Initial work on integrating bit-vector reasoning into iProver.
- Sort support in iProver to facilitate bounded model checking with bit-vectors.
- Improved finite model finding capabilities for iProver.
- We created a Github repository for distributed development of Vampire with our colleagues from TU Vienna and Chalmers University.
- Work started on changing the Vampire SAT and SMT solvers.
- EPR-based k-induction for model checking was developed and integrated into iProver.
- Preliminary work on developing EPR-based methods for the deadlock detection.
- The Vampire workshop on first-order theorem proving and Vampire organised at VSL 2014 (Vienna Summer of Logic), attended by 19 people.
- Preliminary work on options for AVATAR, including the use of third-party SAT solvers, presented at the Vampire Workshop. Vampire was integrated with the SAT solvers MiniSat (Chalmers University) and Lingeling (Johannes Kepler University).
- The first implementation of a concurrent architecture for superposition theorem provers. Several proof attempts can be run concurrently.
- Participation in the annual World Cup in theorem proving (CASC). Vampire is the champion in the main division (FOF). iProver is the champion in two divisions (Satisfiability and EPR). All together Manchester systems won three out of six titles.
- Preliminary work on first-class Boolean type in first-order theorem provers presented at the Vampire workshop.
- Konstantin Korovin gave a joint invited talk at the LaSh’14 and QUANTIFY’14 international workshops on instantiation-based reasoning at the Vienna Summer of Logic (VSL 2014).
- Andrei Voronkov gave a keynote talk at ASE 2014 (Automated Software Engineering).
- Restructuring of iProver code and reworking main data-structures.
- Extending preprocessing and simplification modules in iProver.
- Preliminary work on monadic decomposition for Skolemisation.
- Flexible combination of bounded model checking and k-induction in iProver.
- Integration of non-equivalence constraints into iProver for complete k-induction.

Papers

– Accepted to ATVA 2014: A. Gupta, L. Kovacs, B. Kragl and A. Voronkov, *Extensional Crisis and Proving Identity.*

– N. Bjørner, A. Gurfinkel, K. Korovin and O. Lahav, *Instantiations, Zippers and EPR Interpolation* at Logic for Programming Artificial Intelligence and Reasoning (LPAR’13).

– K. Korovin and M. Veanes, *Skolemization Modulo Theories*. ICMS 2014.

– A paper on EPR-based k-induction is in preparation.

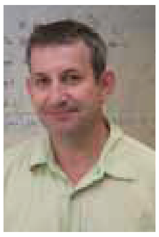
– Paper presented at CAV 2014: A. Voronkov, *AVATAR: a New Architecture for First-Order Theorem Provers* (Andrei Voronkov). It is based on previous results but clearly targets new areas central to REVES, mainly reasoning with both quantifiers and theories.

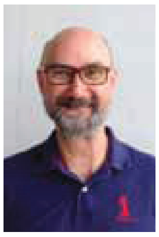
– Abstract published at ASE 2014: Andrei Voronkov. Keynote talk: *EasyChair.*

SeMaMatch: Semantic Malware Matching

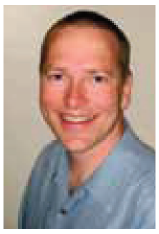
Aims to derive robust semantic signatures for malware classification based on a static and dynamic analysis.

Principal Investigators


Andy King


David Clark

Co-Investigator


Earl Barr

Industrial partner: McAfee Labs

The flood of malware samples is predicted to grow into a deluge in 2012, making the problem of maintaining a database of malware signatures ever more difficult. For each new sample, it is important to determine the threat that it poses. In response to this, dynamic malware analysis tools have been designed that execute the sample in a sandbox, monitoring the actions of a sample. If these actions are similar to those of malware that has been already indexed in the database, then one might draw conclusions regarding provenance and severity of the threat posed. If the sample does not match against known malware, then it can be subject to manual scrutiny, using a dissembler such as IDA Pro.

This Linnaean approach to malware analysis is both natural and convenient: it is natural to group malware into families that share common attributes; and it is provides a convenient way of assessing threat. Yet the whole methodology is predicated on the accuracy with which samples are characterised by their signatures. If a sample is assigned a signature that does not express its behaviour, then samples that are behaviourally distinct can be erroneously grouped together. Conversely, samples which behave the same, but appear different, can be accidentally placed in different groups. The main problem with dynamic malware analysis tools is that they execute the binary for a limited time, typically considering just one path through the binary. This limits the actions that can be observed, rendering the signature inaccurate for programs that reveal their true behaviour later. In addition, the dynamic approach can miss infrequent actions or logic bombs. The dynamic approach is also susceptible to timing attacks that detect a tracer to turn off some action. Above all, the signatures are based solely and only on those actions that are encountered during the trace. More static approaches have been applied too, at one extreme using the call graph of the binary itself for classification, and at the other deploying model checking techniques to search the paths through call graph for signature behaviours that characterise known malware families. Yet graph matching techniques are sensitive to control-flow obfuscation and model checking requires the signature behaviours to be known up-front and distilled into a temporal formula or an automata.

A middle ground is offered by abstract interpretation since it provides a way to systematically consider all paths, while monitoring a program for actions that inform the construction of the signature. Abstract interpretation provides a way to break the dichotomy between the purely dynamic and the purely static approach to malware analysis into a graduated continuum. Formally, purely static approach (a.k.a. a static analysis) can be derived from the purely dynamic approach (a.k.a. a tracer) by compositing a sequence of abstractions. The challenge is to find the hybrid that provides sufficient path coverage to undercover logic bombs yet is sufficiently robust to be used by practitioners in the security sector. The proposed project will discover this sweet point by following two complementary lines of inquiry. Concrete traces will be abstracted to cover more paths and mWEbore actions (at UCL). Static analyses, which cover all paths, will be refined to avoid paths and actions that do not actually occur (at Kent University). Thus UCL will add missing information to signatures (converging on the ideal signature from below) whilst Kent University will remove excess information from signatures (converging on the ideal signature from above). By reflecting on the relative merits of these approaches, we will draw conclusions on how to construct robust signatures for malware classification and thereby advance the whole field.

Key milestones achieved

- Experimented with Normalised Compressed Distance (NCD) for malware matching.
- Experimented with tools for static malware matching based on tree automata.
- Identified problematic experimental results in NCD-based malware matching.
- Investigated graph-edit distance as a metric for malware matching.
- Identified 7z as optimal compressor for NCD mostly due to large compression window.
- Experimented on 600 benign/malware executable binaries using 7z. Mann Whitney Wilcoxon test showed excellent, ordinal separation between the populations using compressibility as a metric. NCD showed almost perfect separation except for two benign programs classed as malware. Interestingly, these were program installers.
- Researched creating 'profiles' for executable binaries – static analysis that identifies high entropy and low entropy regions in the code.
- Developed new incremental algorithm for testing satisfiability of octagons.
- Developed new algorithm for concretising wrapped octagons.
- Developed techniques for combining status register flags with octagons.
- Benchmarked new algorithms and quantified speedup.
- Collected more than 13K Windows executable malware samples from online resources.
- Repeated NCD and compressibility ratio experiments on a larger scale (6000 files). The experiments show that NCD with k-medoid clustering can achieve F-scores of over 0.95. Compressibility ratios, which are cheap to produce (O(n) vs O(n^2) for NCD) achieves F-scores of around 0.85.
- Developed a classifier for malware/benign binary executables on disk using decision forests. Using weak classifiers based only on pairwise NCD we achieved 98% accuracy.
- Experimented with metamorphic engines and NCD. Repeated applications of the engine produced asymptotic upper bounds for some engines. Compressibility rates established clear viability degradation for all programs with repeated engine applications.
- Developed and experimented with a simple algorithm to remove high entropy regions from files and then calculate NCD. High entropy regions are expected to be compressed or encrypted and, therefore, removing these regions might highlight similarities in uncompressed/unencrypted regions of the file. Surprisingly, the results showed that NCD precision/recall is not affected by removing compressed/encrypted regions. However, processing time for NCD was reduced by more than 65%.

Papers and reports

– Aziem Chawdhary, Ed Robbins and Andy King, *Simple and Efficient Algorithms for Octagons*, To appear in APLAS'14, LNCS, Springer-Verlag.

– Ranjeet Singh and Andy King, *Partial Evaluation for Java Malware Detection*, To appear in LOPSTR'14, LNCS, Springer-Verlag.

– William Jones, *On Optimizations for the Volegnant-Jonker Algorithm for the specific task of Graph Edit Distance computation*, Technical Report, September, 2014.

– Nadia Alshawan, Earl Barr, David Clark and George Danezis. *Detecting Malware with Information Complexity and Decision Forests*. Technical Report, September 2014.

– Nadia Alshawan, David Clark, Ibrahim Hadjedjiwa, *Source Code Metamorphism and Information Complexity*, Technical Report, September 2014.

– Kapileshwar Symasundar, *Reverse Engineering Malware with IDA Pro*, September 2014.

Other activities

- Participated in Dagstuhl Seminar 14241 Challenges in Analysing Executables: Scalability, Self-Modifying Code and Synergy, June 9–13 2014.
- UCL second year intern, Kapileshwar Symasundar, funded by GCHQ. Spent the summer reverse engineering malware.
- GCHQ funded a PhD studentship at UCL on finding trigger conditions for malware behaviours using the Conditional Entropy Method.
- Course on Malware added to Information Security MSc at UCL.