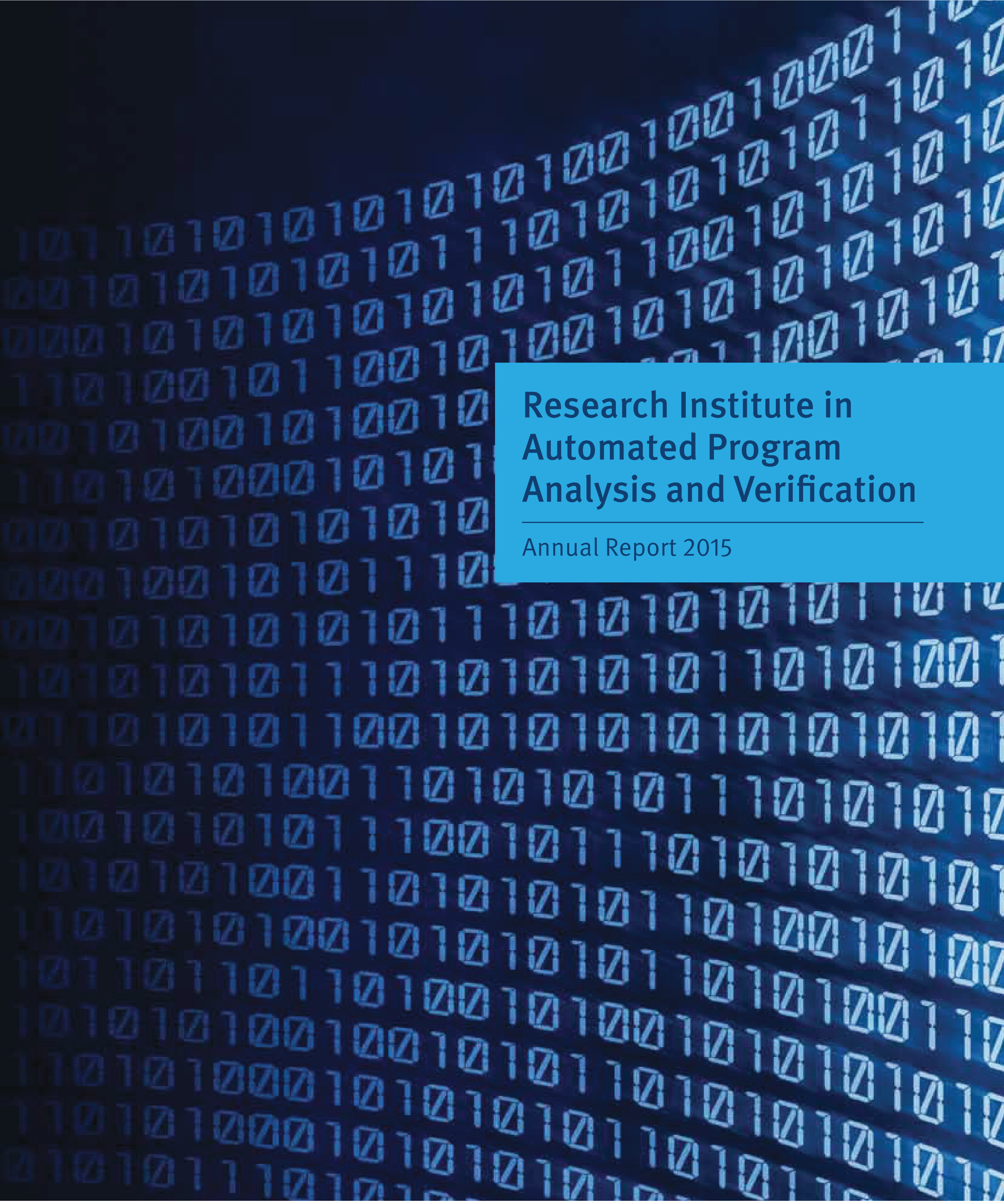


Research Institute in Automated Program Analysis and Verification

Annual Report 2015

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 two years and comprises 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. Impact associated with our projects include:

- The COWL system, which prevents malicious web developers from stealing web users' sensitive information, developed by UCL in collaboration with Mozilla, Google, and Stanford University. The World Wide Web Consortium (W3C), the standards body for the web, has released a draft standard document on COWL, a concrete first step toward deployments in mainstream web browsers like Firefox and Chrome. Open-source release at http://cowl.ws
- A binary fuzzing tool, DexFuzz, developed by Edinburgh and ARM, which fixed major bugs in ART, the new Android runtime, just before the release of Android 5.0. These fixes are now part of the official ART release. Google also modified the DEX specification to remove ambiguity.
- The tool-chain of CBMC, developed at QMUL and Oxford (RI2 advisory board), which has been used to detect more than 700 bugs in open-source software, and is now being integrated into Amazon's software development processes.
- BrowserAudit, an open source tool developed at Imperial to automatically assess the security of web browsers, which has been used to discover previously unknown security issues with the Firefox web browser and the Blackberry mobile browser.
- The theorem prover Vampire, developed by Manchester, which won five divisions out of eight at the annual World Cup of theorem provers CASC. No system in the history had so far won more than three divisions at the same competition. The PI Voronkov received the Herbrand Award for Distinguished Contributions in Automated Deduction.

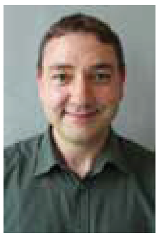
We have held a number of events over the course of this year, including the annual 'Formal Methods and Tools for Security' workshop at Microsoft, and a specialist analysis and verification meeting at GCHQ in Cheltenham attended by approximately 80 people. We have also successfully bid for a Royal Society Scientific Discussion Meeting on 'Verified Trustworthy Software Systems', organised by Gardner with Gordon FRS (Cambridge), Morrisett (Cornell), O'Hearn (Facebook) and Schneider (Cornell). The meeting will lay the groundwork for the next generation of verification grand challenges.

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.

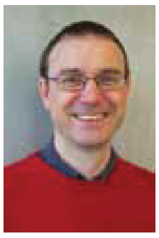
Principal Investigator


David Aspinall

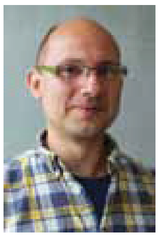
Co-Investigators

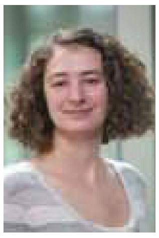
Andrew Gordon

Don Sannella


Ian Stark

Charles Sutton


Björn Franke


Kami Vaniea

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

- We have developed a number of tools for improving Android app security, including:
  - **EviCheck,** which provides 'digital evidence' of security properties that can be used for independent verification and audit. The tool and a tutorial for its use has been made available at: http://groups.inf.ed.ac.uk/security/appguarden/tools/EviCheck
  - **ExplainDroid**, which is a malware classifier that classifies apps as malicious or benign, and provides a human readable explanation of why it thinks a malicious app is bad. We are currently improving ExplainDroid with the help of malware analysts from industry. A technical report on this was produced.
  - **DexFuzz**, made jointly with ARM, which is a binary fuzzing tool that helped fix several major bugs in ART, the new Android runtime, just before the release of Android 5.0. These fixes are now part of the official ART release and based on our bug reports, Google also modified the DEX specification to remove ambiguity.
- We have also devised new probabilistic methods for the longstanding problem in data mining of finding sets of items that occur frequently in a database. These methods hold promise for significantly improved results to many problems in computer security to which association rule mining has already been applied, such as mining of app permissions or security related APIs.
- We have implemented two approaches to generate sound approximations of call graphs for Android apps and designed a way to generate digital evidence asserting the call graph soundness. We made an experimental evaluation showing the pertinence of the new approach with regards to properties related to resource usage in Android.
- We carried out a large scale comparative study between our tool EviCheck and Flowdroid (a state-of-the-art data-flow analysis tool). The goal was to assess the effectiveness and precision of EviCheck. The study involved more than 8,000 apps.
- We studied and developed a method to learn malicious behaviour automata as anti-security policies from instances of human-identified malware families. The algorithm has been implemented and optimised to be able to deal with thousands of behaviour automata which are extracted from apps using static analysis. An extended abstract is being presented at PAS 2015, Program Verification, Automated Debugging and Symbolic Computation, Beijing, China, October 2015 (Chen, Suddon, Gordon, Aspinall, Muttik, Shen).

Papers

- *EviCheck: Digital Evidence for Android* (Seghir, Aspinall) is being presented at 13th International Symposium on Automated Technology for Verification and Analysis, ATVA 2015, Shanghai, China, October 2015.
- *Application of Domain-aware Binary Fuzzing to Aid Android Virtual Machine Testing* (Kyle, Leather, Franke, Butcher, Monteith) in the proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2015.

Other activities

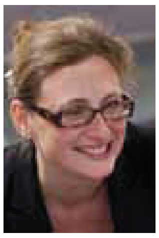
- Work with an undergraduate student to compare several similarity measurement methods and develop a primitive model to cluster apps.
- We hosted two visiting Chinese students from Peking University, who undertook short research projects connected to App Guarden. This was part of an internship scheme to encourage PhD applications from some of the brightest students in China.
- Attendance at FMATS4 meeting in Cambridge by most of the team.

Imperial College London

Certified Verification of Client-Side Web Programs

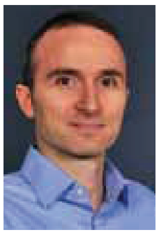
Aims to provide a mechanised specification of the JavaScript standard on which to develop theories and tools for proving correctness, safety and security properties of JavaScript programs.

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.

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

- Last year, we developed JSCert, a mechanised specification of JavaScript which line-by-line follows the ECMAScript 5 English standard, and JSRef, a reference interpreter proved correct with respect to JSCert and tested with the Test262 test suite. This year, improvements to the JSCert project include:
  - Specified and implemented the core Array library functionality in JSCert and JSRef.
  - Completed work on using self-hosted JavaScript from V8 to complete the JSRef implementation of JavaScript Arrays.
  - Generalised JSRef's parser framework, so we can now use any SpiderMonkey-compatible parser (including the popular 'Esprima' parser).
  - Minor bugs discovered and fixed in unverified portions of JSRef.
  - Significantly improved test analysis, coverage and infrastructure.
  - Moved to a 'continuous integration' testing approach.
- A compiler from ECMAScript 5 strict mode to JS-IVL, an intermediate verification language amenable to program analysis. The compiler has been substantially tested using the Test262 test suite, and a core fragment has been proved correct.
- Features added to BrowserAudit to customise test suites and gather results automatically. Fourteen security issues in Blackberry 10 reported to the BlackBerry Security Incident Response Team.
- Change to the ECMAScript 6 standard in how the JavaScript return values interact with while loops by our INRIA partners.
- Organised the ECMAScript committee meeting in Paris with our INRIA partners, and the associated workshop on *Formal Methods Meets JavaScript in Paris* presenting the improved test analysis, the new intermediate verification language for JavaScript, and a new approach to the JSCert project with the aim to place JSCert at the heart of the on-going ECMAScript industrial standardisation effort.

Papers

BrowserAudit: Automated Testing of Browser Security Features (Hothersall-Thomas, Maffeis, Novakovic), International Symposium on Software Testing and Analysis, July 2015.

A Trusted Mechanised Specification of JavaScript: One Year On (Invited Keynote and Paper) (Gardner, Smith, Watt, Wood), International Conference on Computer Aided Verification, July 2015.

ES5strict $\rightsquigarrow$ IVL, Principled Translation using Operational Semantics (Naudžiūnienė), Workshop on Tools for JavaScript Analysis (JSTools) co-located with ECOOP, July 2015.

Hybrid Typing of Secure Information Flow in a JavaScript-like Language (Santos, Jensen, Rezk, Schmitt), Symposium on Trustworthy Global Computing, August 2015.

Modular Monitor Extensions for Information Flow Security in JavaScript (Santos, Rezk, Matos), Symposium on Trustworthy Global Computing, August 2015.

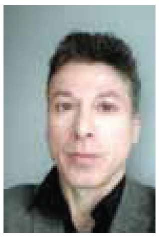
Other activities

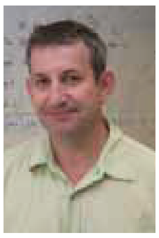
- Gardner successfully led the bid for a Royal Society meeting on 'Verified Trustworthy Software Systems' with Gordon FRS, Morrisett, O'Hearn and Schneider, April 2016.
- Gardner organised a Dagstuhl meeting on 'Compositional Methods for Next-generation Concurrency' with Birkedal, Dreyer and Shao.
- Gardner delivered a new undergraduate and MSc course on 'Modular Reasoning about Programs', and Maffeis and Novakovic (with Huth) delivered a new undergraduate and MSc course on 'Networks and Web Security' at Imperial.
- Maffeis with Novakovic successfully bid for a GCHQ small grant to set up a number of machines and built a server for experimenting with network and web security.
- Novakovic gave the talk 'BrowserAudit: Automated Testing of Browser Security Features' at FMATS4.
- BrowserAudit was additionally submitted as an artifact to the 2015 International Symposium on Software Testing and Analysis, where it was judged by the Artifact Evalution Committee to have exceeded expectations; Novakovic also gave an invited talk on *BrowserAudit* in the tool demonstration track.

Compositional Security Analysis for Binaries

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

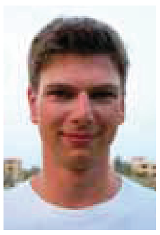
Principal Investigators


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

We developed an analysis, based on clear security principles and verification tools, which is automatic and effective in detecting information leaks. We analysed all of OpenSSL for absence of confidentiality leaks of similar nature to Heartbleed. This is possibly the largest automated information leakage analysis of OpenSSL and more generally the first automatic leakage analysis for real-world complex software written in C. To extend this from C programs to binaries, we have developed a certified type-based decompiler and used this to recover a strongly typed C-like program from x86 executables.

- Formalised type and argument recovery by formalising high-level and low-level semantics, and the (decompilation) relationship between them.
- CBMC security analysis of Heartbleed bug and other OpenSSL functions.
- Developed new generic infrastructure for control-flow recovery and instruction decoding.
- Infrastructure to perform static analysis on the entire Debian/Linux distribution (approximately 450 million lines of C code, more than one billion lines of source code in total).
- Developed translation relation for formalising decompilation.
- Developed the concept of a high-level witness so as to justify a low-level type assignment to a binary executable.
- Information-leak analysis of OpenSSL, including Heartbleed.
- Improved algorithmic for information flow analysis using SMT-solvers and quantifiers.
- Finalised formalisation of type and argument recovery.

Papers

Simple and Efficient Algorithms for Octagons (Aziem Chawdhary, Edward Robbins and Andy King), in proceedings of the 12th Asian Symposium on Programming Languages and Systems (APLAS).

Automating Software Analysis at Large Scale (Daniel Kroening, Michael Tautschnig), in post-proceedings of MEMICS 2014.

All-Solution Satisfiability Modulo Theories: applications, algorithms and benchmarks (Quoc-Sang Phan and Pasquale Malacaria), in proceedings of the International Conference on Availability, Reliability and Security (ARES) 2015.

From MinX to MinC: Semantics-Driven Decompilation of Recursive Datatypes (Ed Robbins, Andy King and Tom Schrijvers), Principles of Programming Languages, ACM Press, 2016.

Other activities

- Completed ELF loader and amalgamated with CFG recovery.
- On-going development of IC3 and integration with CBMC.
- GCHQ supported infrastructure for large-scale benchmarking.
- Combined IC3 with symbolic execution.
- Presentation and participation in FMATS4.
- Preparation of tutorial for GCHQ on CBMC, pending agreed date for presentation.

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

- COWL (Confinement with Origin Web Labels), a label-based mandatory access control system for web browsers (written with collaborators at Google, Mozilla, and Stanford) presented at OSDI 2014, the co-premier venue for computer systems research (held every other year, alternating with SOSP).
- PI Karp presented COWL to Microsoft's research and web browser development teams at Microsoft's Redmond, Washington headquarters on 20 April 2015, and at the 4th Workshop on *Formal Methods and Tools for Security* (FMATS4) at Microsoft Research Cambridge on 11 June 2015.
- Initial design and implementation of a source-to-source JavaScript compilation realisation of the G8 security policy system for Node.js. G8 allows programmers to express simple, powerful security policies for Node.js applications using labelled information flow control, and enforces these policies at run-time.
  - Completed implementation of all three main compiler passes of the JavaScript-to-JavaScript compiler to compile G8 functionality into core JavaScript code within Node.js applications.
  - G8 source-to-source compiler tested to check that (a) transformed code output by compiler is as expected, (b) transformed code produces expected output when run under V8_ (c) transformed code behaves as expected on simple examples, propagating labels successfully across simple operations and data flows.
- Initial design and implementation of a high-performance, least-privilege-isolated web server that uses Software-Based Fault Isolation (SFI) rather than Linux processes to enforce memory isolation between program compartments.
- Design and implementation of a new program analysis tool that proves properties of a program that no known tool to date could automatically prove, of the form: "It is possible that in the future the event X could occur infinitely often." Such properties are useful as part of establishing the practical security of real-world programs.

Papers

Spatial Interpolants (Albargouthi, Berdine, Cook and Kincaid) in ESOP 2015.

Fairness for Infinite-State Systems (Cook, Khlaaf and Piterman) in TACAS 2015.

On Automation of CTL Verification for Infinite-State Systems* (Cook, Khlaaf and Piterman) in CAV 2015.

Other activities

- Positive press coverage on COWL in high-profile technical news outlets, including:

  Network World
  www.networkworld.com/article/2691741/microsoft-subnet/researchersunveil-cowl-a-new-system-to-protect-surfers-privacy.html

  The Register
  www.theregister.co.uk/2014/10/07/boffins_build_cowl_web_privacy_system_to_cut_malware_off_at_the_knees

  Engadget
  www.engadget.com/2014/10/06/cowl-web-privacy
- Presented COWL at UK Cyber Security Research Conference 2014 (hosted by BIS); COWL mentioned as example of RI1 and RI2 success in welcome speech by James Quinault CBE, Director of the Office for Cyber Security and Information Assurance, Cabinet Office.
- Progress standardising COWL through the World Wide Web Consortium (W3C): imminent October 2015 release of First Public Working Draft (FPWD) standard for COWL.
- PI Karp served as programme co-chair of the ACM SIGCOMM 2015 conference, held in London in August 2015.
- After a year and a half of excellent contributions to the G8 security policy system for Node.js, RA Katrina Joyce has left the project to take up a position as a security engineer at Google's London office, where she will be part of a small, elite security research and development team reporting to Ben Laurie.
- PI Karp invited to give a keynote address at ACM SYSTOR 2016.

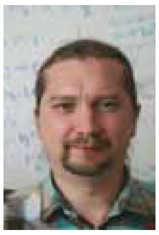
REVES: REasoning in VErification and Security

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

- Our theorem prover Vampire won five divisions out of eight at the annual World Cup of theorem provers CASC. No system in the history had so far won more than three divisions at the same competition.
- The project PI Andrei Voronkov received the Herbrand Award for Distinguished Contributions in Automated Deduction.
- The AVATAR architecture was improved considerably: many options implemented and tested, several SAT solvers embedded and tested. MiniSAT was eventually chosen as the Vampire SAT solver, bug-fixed and improved. As a result, the new releases of Vampire are considerably stronger than previous ones.
- Satisfiability checking in Vampire was re-implemented. Finite model building, including new kinds of symmetry breaking for functions implemented. As a result, Vampire has become number one first-order prover in both first-order satisfiability checking and nearly propositional reasoning.
- A new method of theory reasoning for superposition provers was developed and implemented in Vampire. As a result, Vampire has become number one first-order prover in reasoning with quantifiers and theories.
- We introduced logic FOOL having a first-class Boolean type, if-then-else and let-in constructs. The logic bridges the gap between constructs used in programming languages and first-order logic, and so makes first-order theorem provers more useful for program analysis and other applications. FOOL was implemented in Vampire and tested. We also implemented a new rule (FOOL paramodulation) for reasoning with Boolean types in superposition provers and carried out experiments showing its efficiency.
- Implementation of polymorphic arrays in Vampire.
- Integration in Vampire of the SMT solver Z3.
- Second Vampire Workshop organised at CADE 2015, again featuring more than 20 participants.
- Vampire solved an open problem in semi groups thanks to the use of the AVATAR architecture.
- We developed verification techniques for bounded and unbounded model checking based on encodings into the effectively propositional fragment of first-order logic (EPR). In particular, we developed a novel framework (UCM) for EPR-based bounded model checking and k-induction based on counter-example guided abstraction-refinement.
- We integrated the UCM framework for BMC and k-induction into the iProver system with a range of optimisations.
- We extended UCM with reductions based on cone of influence and integrated them into iProver.
- AIG verification format integrated into iProver.
- Instantiation-based model representation developed for efficient abstraction-refinement.
- First-order lemmas and invariant generation in the UCM framework.
- Improved integration of first-order sorts into iProver.
- Developed incremental satisfiability checking in iProver.
- Extended and improved main data-structures, indexing, pre-processing and simplifications in iProver.

Papers

*Cooperating Proof Attempts (*Giles Reger, Dmitry Tishkovsky, Andrei Voronkov), CADE 2015 (International Conference on Automated Deduction): 339–355.

Playing with AVATAR (Giles Reger, Martin Suda, Andrei Voronkov), CADE 2015 (International Conference on Automated Deduction): 399–415.

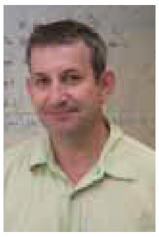
A First Class Boolean Sort in First-Order Theorem Proving and TPTP (Evgenii Kotelnikov, Laura Kovács, Andrei Voronkov), CICM (Conference on Intelligent Computer Mathematics) 2015: 71–86.

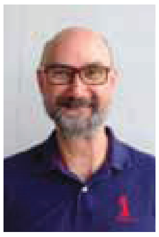
EPR-based k-induction with Counterexample Guided Abstraction Refinement (Zurab Khasidashvili, Konstantin Korovin and Dmitry Tsarkov), GCAI (Global Conference on Artificial Intelligence) 2015: 133–146.

SeMaMatch: Semantic Malware Matching

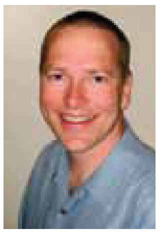
Aims to derive robust semantic signatures for malware classification based on a static and dynamic analysis.

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

- Investigated VMs in which to execute and collect trace information.
- Generated a set of virus families using NGVCK.
- Applied decision forest learning algorithm to improve efficiency of malware detection using NCD.
- Compared accuracy rates against malware engines hosted by VirusTotal.
- Improved the Volgenant-Jonker algorithm for approximating GED.
- Applied the algorithm to compare the distortion between the CFG of two binaries.
- Translated x86 binaries into RReil intermediate for CFG generation by abstract interpretation.
- Developed software for wavelet analysis of strings with the aim of identifying regions of high and low entropy and patterns based on these.
- Used PIN to instrument execution of binaries and collect trace data.
- Abstracted op code traces and compared NCD based classification for binaries, and op code traces.
- Applied the Volgenant-Jonker algorithm for approximating GED.
- Developed an unbounded model-checker to derive call-graph dependencies.
- Partial Evaluation for Java Malware Detection.
- Detecting Malware with Information Complexity.
- Investigated structural entropy as the basis of a similarity metric on binary executables.
- Developed language models for malware and benign executables.
- Investigated partial evaluation of Jimple bytecode.
- Developed a new O(max(n, m)) graph matching approximation algorithm of two graphs with n and m nodes.

Papers

Revisiting Volgenant-Jonker for Approximating Graph Edit Distance (William Jones, Aziem Chawdhary, Andy King) (2015), in: Liu, C. -L. *et al*. eds. Graph-based Representations in Pattern Recognition. Springer, pp. 98–107.

Partial Evaluation for Java Malware Detection (Ranjeet Singh, Andy King) (2015), in: Proietti, Maurizio and Seki, Hirohisa, eds. Twenty fourth International Symposium on Logic-Based Program Synthesis and Transformation. Lecture Notes in Computer Science, 8991 (a longer version by Ranjeet Singh, Aziem Chawdhary and Andy King has been submitted to Formal Aspects of Computing, pending acceptance).

Simple and Efficient Algorithms for Octagons (Aziem Chawdhary, Edward Robbins, Andy King), 296–313 Programming Languages and Systems, APLAS 2014, Singapore, Lecture Notes in Computer Science 8858, Springer 2014.

Detecting Malware with Information Complexity (Nadia Alshahwan, Earl T. Barr, David Clark, George Danezis) http://arxiv.org/abs/1502.07661

Other activities

- Talk and participation at FMATS4 in Cambridge.
- Hector David Menendez was appointed as RA at UCL.