ABSTRACTS

Verified trustworthy software systems

Wednesday 6 – Thursday 7 April 2016

Imperial College London

Peter Sewell Computer Laboratory, University of Cambridge, UK	**Wednesday 6th April, Session 1, 09:00-09:30** Rigorous Engineering - What Can We Do? What Should We Do? *Peter Sewell, Computer Laboratory, University of Cambridge, UK* The REMS project, Rigorous Engineering for Mainstream Systems, is developing and applying semantic models and tools for several key abstractions, including ARM, IBM POWER, and CHERI multiprocessors, C/C++ concurrency, C language, linking, POSIX filesystems, and TLS. This talk will summarise some of the work so far, focussing on the many different ways such models can be useful and on the different modes of interaction with system builders and systems researchers that we have found to be effective.
Warren Hunt The University of Texas, USA	**Wednesday 6th April, Session 1, 09:30-10:00** Specification and Verification of x86 machine-level code. *Warren Hunt, The University of Texas, USA* We are using theorem-proving techniques to model and analyze x86 software for the purpose of increasing the accuracy and reliability of x86-based products. We have developed an ISA-level x86 emulator in the ACL2 logic; this emulator serves as a precise specification for x86 software. Using the ACL2 theorem-proving system, we describe how our x86 ISA model is used to prove the correctness of x86 binary-level programs.
Anna Slobodova Centaur Technology, Inc, USA	**Wednesday 6th April, Session 1, 10:00-10:30** Formal Analysis of x86 micro-processor design at Centaur Technology. *Anna Slobodova, Centaur Technology, Inc, USA* Centaur Technology has adopted formal verification (FV) as a part of their production flow. Our company designs Intel compatible x86-64 microprocessors. To assure the quality of our design, a lot of effort is spent on validation. Recently, additions to x86-64 architecture widened the data on which instructions are performed; thus simulation provides even less coverage with respect to all possible inputs to the system than just a few years ago. The FV team at Centaur Technology was created as a reaction to this trend as well as the fact that the capacity of formal tools has reached a level where they can be successful used even on industrial-scale designs. Our team has worked on a variety of design and verification problems including microcode verification and transistor-level validation, but our main focus remains the verification of register-transfer-level designs written in System Verilog. Our verification framework is built on top of the ACL2 theorem prover. There are many decision procedures built in the logic of ACL2 and proved correct within this logic. For example, packages exist, are defined and verified inside the logic, for Binary Decision Diagrams (BDD) and for And-Inverter Graph (AIG) manipulation. The symbolic simulator, the core of our verification methods, is also verified.

Andrei Sabelfeld

Chalmers University of Technology, Sweden

Wednesday 6th April, Session 1, 11:00-11:30

Reconciling the Clash of Security Goals: Websites vs. Browser Extensions.
Andrei Sabelfeld, Chalmers University of Technology, Sweden

Browser extensions provide a powerful platform to enrich browsing experience. At the same time, they raise important security questions. From the point of view of a website, some browser extensions are invasive, removing intended features and adding unintended ones as, e.g., extensions that hijack Facebook likes. Conversely, from the point of view of extensions, some websites are invasive as, e.g., websites that bypass ad blockers. We illuminate the tension of the security goals by several empirical studies of the state of the art in browser extension security. Driven by the "users > developers > browser" philosophy, we discuss measures to reconcile the conflicting security goals and propose policies and mechanisms to improve the state of the art.

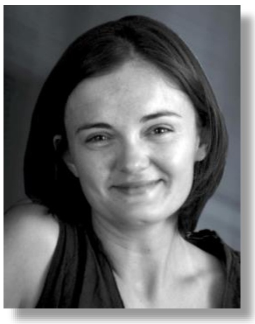
Lujo Bauer

Carnegie Mellon University, USA

Wednesday 6th April, Session 1, 11:30-12:00

Accurate, Robust Metrics for Usable Security.
Lujo Bauer, Carnegie Mellon University, USA

A critical part of building secure systems is verifying that they're usable and secure in practice, and not just theoretically. This requires accurate and robust metrics for quantifying this "usability" and "practical security". For several years, we've been studying how to help users create passwords that are hard for attackers to crack, but are still easy for users to remember and use. A key challenge in this work was to develop and validate a methodology for collecting passwords and assessing their strength and usability. I'll discuss our approach, and how we applied it to over 50,000 participants to study the effects of password-composition policies, password-strength meters, and detailed, step-by-step feedback and guidance during the password creation policies.



Alexandra Silva

University College London UK

Wednesday 6th April, Session 1, 12:00-12:30

Probabilistic NetKAT.
Alexandra Silva, University College London, UK

In this talk, we will present a new language for network programming based on a probabilistic semantics. We extend the NetKAT language with new primitives for expressing probabilistic behaviors and enrich the semantics from one based on deterministic functions to one based on measurable functions on sets of packet histories. We establish fundamental properties of the semantics, prove that it is a conservative extension of the deterministic semantics, show that it satisfies a number of natural equations, and develop a notion of approximation. We present case studies that show how the language can be used to model a diverse collection of scenarios drawn from real-world networks. This is joint work with Nate Foster, Dexter Kozen, Konstantinos Mamouras and Mark Reitblatt.

Derek Dreyer

Max Planck Institute for Software Systems (MPI-SWS), Germany

Wednesday 6th April, Session 2, 14:00-14:30

RustBelt: Logical Foundations for the Future of Safe Systems Programming.

Derek Dreyer, Max Planck Institute for Software Systems (MPI-SWS), Germany

Rust is a new language developed at Mozilla Research that marries together the low-level flexibility of modern C++ with a strong "ownership-based" type system guaranteeing type safety, memory safety, and data race freedom. As such, Rust has the potential to revolutionize systems programming, making it possible to build software systems that are safe by construction, without having to give up low-level control over performance. Unfortunately, none of Rust's safety claims have been formally investigated, and it is not at all clear that they hold. To rule out data races and other common programming errors, Rust's core type system prohibits the aliasing of mutable state, but this is too restrictive for implementing some low-level data structures. Consequently, Rust's standard libraries make widespread internal use of "unsafe" blocks, which enable them to opt out of the type system when necessary. The hope is that such "unsafe" code is properly encapsulated, so that Rust's language-level safety guarantees are preserved. But due to Rust's bleeding-edge type system, along with its reliance on a weak memory model of concurrency, verifying that Rust and its libraries are actually safe will require fundamental advances to the state of the art. In the RustBelt project, funded by an ERC Consolidator Grant, we aim to equip Rust programmers with the first formal tools for verifying safe encapsulation of "unsafe" code. To achieve this goal, we will build on recent breakthrough developments in concurrent program logics and semantic models of type systems.

Alastair Donaldson

Imperial College London, UK

Wednesday 6th April, Session 2, 14:30-15:00

Metamorphic Compiler Testing.

Alastair Donaldson, Department of Computing, Imperial College London, UK

The correctness of compilers is of key importance to software engineers, and an implicit assumption of compiler correctness underpins many claims of soundness made about program analysis techniques that operate on source code. Compiler testing and verification has received a lot of research attention over the last few decades, with two recent notable achievements being the Csmith tool for fuzz-testing C compilers, and the CompCert verified compiler. A recent paper (a distinguished paper at PLDI'14), "Compiler Validation via Equivalence Modulo Inputs", by Le, Afshari and Su, proposed a new compiler testing approach based on metamorphic testing (a term coined by T.S. Chen), whereby compiler reliability is examined by comparing results for compiled executables arising from multiple programs that, while structurally different, should guarantee producing identical outputs for certain known inputs. Le et al. derived metamorphic relations by using profiling to identify regions of code in a program not covered by a given input I, producing variants of the program obtained by mutating these "I-dead" regions. Such variants should behave identically when executed on input I. Inspired by this work, in our recent paper, "Many-Core Compiler Fuzzing" (Lidbury, Lascu, Chong, Donaldson, PLDI'15), we investigated the injection of "dead-by-construction" code into existing test cases, in effect manufacturing "I-dead" code through the use of opaque predicates. We found this to be effective in uncovering wrong code compiler bugs, and it is attractive because it does not require the a priori existence of I-dead code, nor of an available profiler to discover it (which may not be available, say, in the context of compilers for GPU computing). I will give an overview of this existing work, and then focus on our ongoing work in which we are investigating more aggressive uses of opaque predicates for metamorphic compiler testing. I will also argue that metamorphic techniques are particularly effective in testing compilers in the domain of computer graphics, where floating point semantics are deliberately under-specified to support implementation differences, and where there is thus no absolute notion of the correct result for a compiler test case.

Chris Hawblitzel

Microsoft Research, USA

Wednesday 6th April, Session 2, 15:00-15:30

IronFleet: Proving Practical Distributed Systems Correct.

Chris Hawblitzel, Microsoft Research, USA

I will describe our experience building IronFleet, a collection of verified distributed systems. Using a combination of TLA-style state-machine refinement and Hoare-logic verification, we verified both a complex implementation of a Paxos-based replicated state machine library and a lease-based distributed key-value store. We proved that each obeys a concise safety specification, as well as desirable liveness requirements. Our verified implementations achieved performance reasonably competitive with existing unverified systems. We wrote IronFleet in Dafny, which uses the Z3 automated theorem prover to prove verification conditions. This does not mean, however, that the verification was completely automatic. I will briefly discuss how much automation the verification of various parts of the system enjoyed, reasons for successful automation (or lack thereof), and what sort of manual effort was required when automation fell short.

Five minute talks: Post-doctoral researchers and PhD students will present 5-minute snapshots of their research.

Wednesday 6th April, Session 2, 16:00-16:30

Micro-Policies: Formally Verified, Tag-Based Security Monitors.

Arthur Azevedo de Amorim, University of Pennsylvania, USA

Let's Face It: Faceted Values for Taint Tracking.

Daniel Schoepe, Chalmers University of Technology, Sweden

End-to-end verification of Concurrent Separation Logic through an optimizing compiler to a weak-cache-consistent computer.

Santiago Cuellar, Princeton University, Princeton, USA

Compilation Using Correct-by-Construction Program Synthesis.

Clément Pit-Claudel, Massachusetts Institute of Technology, USA

Formalising POSIX File Systems.

Gian Ntzik, Imperial College London, UK

Operational Aspects of C/C++ Concurrency.

Anton Podkopaev, St. Petersburg State University, St. Petersburg, Russia

Leakage Aware Verification

David McCann, Bristol University, UK

Protocol decomposition for security and verifiability.

Vincent Rahli, University of Luxembourg, Luxembourg

JSLINQ: Building Secure Applications across Tiers,

Musard Balliu, Chalmers University of Technology, Sweden

ELF, linking, ABIs, and all that

Dominic Mulligan, Computer Laboratory at the University of Cambridge. UK (joint work with Stephen Kell and Peter Sewell) Automatic Verification of Security and Privacy in

Cryptographic Protocols using Temporal-Epistemic Logic

Ioana Boureanu , Imperial College London, UK, Marie Skłodowska-Curie (MSCA)

Verifying probabilistic systems

Ben Sherman, Massachusetts Institute of Technology, CSAIL, USA

Phantom Monitors: A Simple Foundation for Modular Proofs of Fine-Grained Concurrent Programs

Christian Bell, Massachusetts Institute of Technology, CSAIL, USA

Dr Jérôme Feret

INRIA, France

Wednesday 6th April, Session 2, 17:00-17:30

An overview of the Astrée/AstréeA analyzer.

Jérôme Feret, INRIA, France

Astrée is a static program analyzer aiming at proving the absence of Run Time Errors (RTE) in programs written in the C programming language. Astrée targets structured C programs with complex memory usages or even dynamic memory allocation, but without recursion. AstréeA is an extension of Astrée that to deal with multi-threaded software satisfying the ARINC 653 specification. Astrée was able to prove completely automatically the abscence of any RTE in the primary flight control software of the Airbus A340 fly-by-wire system and of the Airbus A380, and the automatic software of the Jules Vernes Automated Transfer Vehicle. AstréeA has been used to analyze modules of differing criticality levels under the Integrated Modular Avionics (IMA) airborne system. In this talk, after a short reminder of the Astrée story, we give an overview of the structure of the analyzer and describe some of the abstractions that are used within the analyzer.

John Launchbury

DARPA, USA

Wednesday 6th April, Session 2, 17:30-18:00

How Far Are We?

John Launchbury, Director I2O, DARPA, USA

Formal methods are making tremendous technical progress and having an impact in multiple application areas. So, what gaps remain? In this talk we will look at some techniques for enhancing the reach of formal methods to address issues like cyber security in complex systems, but will also identify major challenges and opportunities that remain.

Professor Gernot Heiser

University of New South Wales, Australia

Thursday 7th April, Session 1, 09:00-9:30

OS Kernel Design for Verification – Security is No Excuse for Bad Performance.

Gernot Heiser, University of New South Wales, Australia

The seL4 microkernel is known for its formal proof of implementation correctness and security enforcement, but it is also the world's fastest general-purpose microkernel. This is the result of design and implementation carefully balancing the potentially conflicting requirements of verifiability and low overhead — from day one.
In this talk I will look at some case studies of such verification-oriented high-performance design, specifically for supporting multicore processors and temporal isolation. Our experience shows that trying to make two masters happy (verification and performance) requires some thought, but is possible and arguably leads to better designs than when trying to serve only one master. The core technical learning is that strict adherence to the microkernel minimality principle is a key to success.

Adam Chlipala

MIT, SAIL, USA

Thursday 7th April, Session 1, 9:30-10:00

Bedrock and Fiat: Specifications and Proofs at the Center of a Programming Ecosystem.

Adam Chlipala, MIT, SAIL, USA

We know how to build rich ecosystems of programming tools, where each part of a project can be coded in the language most suited to it, and yet all of the parts cooperate together in the final running system. The central question of this meeting is how we come up with not just code but also proofs that it meets specifications. How should our programming-tool ecosystems change to support that goal? Which common foundations should be fixed across tools, and which design patterns apply well to particular kinds of tools? Can we leverage formal specifications to improve the programming process even independently of verification? I'll present our work addressing all of those questions in one prototype platform, based on the Coq libraries Bedrock and Fiat. At the heart of the system is the equivalent of an object-file format that includes specifications and proofs. All code in the system must eventually be translated to this format, and all formal arguments must eventually be phrased in terms of it. However, many different programming and verification tools can get us there, from whichever source languages we like. By committing to "formal specifications everywhere," we can even program at higher abstraction levels than we're used to, starting from the specification and deriving the program automatically. The approach that we take there also provides another perspective on the foundations of domain-specific programming languages. I will introduce the core of the system and several case studies applying it to construct full verified programs, including with the first automated proof-generating pipeline from relational specifications to fast assembly code.

Georges Gonthier

Microsoft Research, USA

Thursday 7th April, Session 1, 10:00-10:30

Interactive Theorem Proving.

Georges Gonthier, Microsoft Research, USA

Cristian Cadar

Imperial College London, UK

Thursday 7th April, Session , 10:00-11:30

Multi-version execution for improved software reliability and security.

Cristian Cadar, Imperial College London, UK

In this talk, I would give an overview of our ongoing work on designing multi-version execution techniques, in which multiple versions or variants of the software are run in parallel in order to improve the reliability and security of deployed software systems. I will first show how a multi-version execution framework can be effectively designed by combining selective binary rewriting with a novel event-streaming architecture to significantly reduce performance overhead and scale well with the number of versions. I will then describe how multi-version execution can be used to improve software updates, deploy expensive dynamic analyses in production code, and detect compiler backdoors in security sensitive programs.

Heiko Mantel

TU Darmstadt, Germany

Thursday 7th April, Session 1, 11:30-12:00

Information-Flow Security for Concurrent Programs: Pitfalls, Solutions, and Challenges.

Heiko Mantel, TU Darmstadt, Germany

Today's IT-systems store and process an abundance of data. Some of this data represents private or secret information, and a crucial question is how to ensure that no such information is leaked. We focus on techniques that establish confidentiality guarantees by controlling the flow of information within IT-systems. Research on information-flow security has resulted in an impressive portfolio of program analysis techniques and tools. For sequential programs, these approaches are becoming rather mature, and a good understanding of the resulting, declarative security guarantees has been achieved. For concurrent programs, information-flow security is more challenging. The first part of the talk provides an introduction to theory and practice of information-flow control. The second part clarifies why it is so hard to achieve information-flow security for concurrent programs, and discusses recent results from our on-going research effort to tackle these challenges.

Pasquale Malacaria

Queen Mary University of London, UK

Thursday 7th April, Session 1, 12:00-12:30

Information leakage analysis of complex C code and its application to OpenSSL.

Pasquale Malacaria, Queen Mary University of London, UK

The worldwide attention generated by the Heartbleed bug has demonstrated even to the general public the potential devastating consequences of information leaks. While substantial academic work has been done in the past on information leaks, these works have so far not satisfactorily addressed the challenges of automated analysis of real-world complex C code. On the other hand, effective working solutions rely on ad-hoc principles that have little or no theoretical justification. The foremost contribution of this work is to bridge this chasm between advanced theoretical work and concrete practical needs of programmers developing real world software. We present an analysis, based on clear security principles and verification tools, which is largely automatic and effective in detecting information leaks in complex C code running everyday on millions of systems worldwide. (joint work with Michael Tautschnig and Dino Distefano)

Mark S. Miller

Google

Thursday 7th April, Session 2, 14:00-14:30

Frozen Realms: draft standard support for safer JavaScript plugins.

Mark S. Miller, Google

Arthur Charguéraud

INRIA, France

Thursday 7th April, Session 2, 14:30-15:00

An interactive debugger for the JavaScript specification.

Arthur Charguéraud, INRIA, France

In the recent years, it has been shown that JavaScript can be given a formal semantics faithful to the standards (EcmaScript, version 5), and that such a formal semantics can be executed on test programs. In this talk, I will describe ongoing efforts for making such a formal semantics more likely to be eventually adopted as the reference semantics by the JavaScript committee. In particular, I will present an interactive debugger for executing the JavaScript specification on concrete programs, allowing to visualize both the state of the interpreter and the state of the interpreted program. I will also describe on-going work on the automated generation of pretty-big-step evaluation rules from the monadic code of the interpreter, and speculate on the generation of English prose from the code of the interpreter.

Viktor Vafeiadis Max Planck Institute for Software Systems (MPI-SWS), Germany	**Thursday 7th April, Session 2, 15:00-15:30** Validating optimizations of concurrent C/C++ programs. *Viktor Vafeiadis, Max Planck Institute for Software Systems (MPI-SWS), Germany* We present a validator for checking the correctness of LLVM compiler optimizations on C11 programs as far as concurrency is concerned. Our validator checks that optimisations do not change memory accesses in ways disallowed by the C11 and/or LLVM memory models. We use a custom C11 concurrent program generator to trigger multiple LLVM optimisations and evaluate the efficacy of our validator. Our experiments highlighted the difference between the C11 and LLVM memory models, and uncovered a number of previously unknown compilation errors in the LLVM optimisations involving the C11 concurrency primitives.
Magnus Myreen Chalmers University of Technology, Sweden	**Thursday 7th April, Session 2, 15:30-16:00** The CakeML project: overview and a new compiler backend. *Magnus Myreen, Chalmers University of Technology, Sweden* CakeML is a strict functional programming language in the style of Standard ML and OCaml. It has a verified implementation with verified parsing, type inference, and compilation. I will talk about a recent reimplementation of the compiler's backend. The new backend is designed to resemble mainstream (unverified) compilers. In particular, the new compiler uses a number of intermediate languages that allows it to incrementally compile away high-level features and enables verification at the right levels of semantic detail. The new compiler supports efficient curried multi-argument functions, configurable data representations, register allocation, efficient stack representations, and several target architectures: x86-64, ARMv6, ARMv8, MIPS-64, and RISC-V. The CakeML project is a collaboration between people at Chalmers (Sweden), Data61/NICTA (Australia), Cambridge (UK), A*STAR IHPC (Singapore) and Kent (UK).