THE FOURTEENTH ANNUAL
HIGH CONFIDENCE
SOFTWARE AND SYSTEMS
CONFERENCE
Annapolis, MD | May 6-8, 2014
http://cps-hcss.org
The Fourteenth Annual

High Confidence
Software and Systems
Conference

Annapolis, Maryland, USA | May 6-8, 2014
http://cps-hcss.org

Program Co-Chairs
Byron Cook, MS Research & University College London
John Hatcliff, Kansas State University

Steering Committee
Kathleen Fisher, DARPA
John Launchbury, Galois, Inc.
Brad Martin, National Security Agency
Ray Richards, Rockwell Collins
Bill Scherlis, Carnegie Mellon University
The Co-Chairs and steering committee are pleased to welcome you to the 14th annual High Confidence Software and Systems (HCSS) Conference being held again this year at the Historic Inns of Annapolis in Annapolis, Maryland.

This year’s program continues the tradition of excellence over the fourteen-year history of the Conference. A host of world class research scientists representing academia, industry, and Government will deliver a range of experience and technical talks structured to focus on new scientific and technological foundations that can enable entirely new generations of engineered designs that are becoming essential for effectively operating life-, safety-, security-, and mission-critical systems. New foundations in science, technology, and advanced practice continue to be needed to build these systems with computing, communication, information, and control pervasively embedded at all levels. Talks will be focused on the themes of Requirements and Specification, Mobility, Reasoning about Control, and Designed-In Security. These themes and other topics will also be depicted through technical poster displays at this year’s poster session.

We are pleased to host the Software Certification Consortium (SCC) meeting again this year. Formed in 2007, the SCC comprises industry researchers, government regulators, and academicians whose goal is to understand certification issues with respect to systems that contain significant software components (e.g., aerospace, automotive, medical devices, nuclear, defense, etc.), and to objectively make recommendations on processes and standards that impact the certification of such systems.

We hope that you will find the 2014 Conference as stimulating and informational as in years past. We greatly appreciate your attendance, and look forward to your continued participation and support of future conferences.

Sincerely,

Byron Cook  
Microsoft Research  
University College London

Kathleen Fisher  
DARPA

Brad Martin  
National Security Agency

William Scherlis  
Carnegie Mellon University

John Hatcliff  
Kansas State University

John Launchbury  
Galois, Inc

Raymond Richards  
Rockwell Collins
# TABLE OF CONTENTS

Welcome Message ................................................................. 2

Table of Contents ..................................................................... 3

General Information ............................................................... 4

Conference Organization ......................................................... 5

Program Agenda .................................................................... 6

Conference Presentations ....................................................... 9

Poster Session Presentations ................................................. 47

Conference Dinner ............................................................... 62

Local Restaurants ............................................................... 63

Notes .................................................................................... 64
GENERAL INFORMATION

REGISTRATION
Registration will be located in the state lobby of the Governor Calvert House and will be open:

- 11:00 a.m. to 5:00 p.m. Sunday
- 8:00 a.m. to 5:00 p.m. Monday through Thursday

WIRELESS INTERNET CONNECTION
A wireless Internet connection will be available in the Governor Calvert Ballroom and Atrium. The network name is: **Governor Calvert**, and the password is **Flower**.

POSTER PRESENTATIONS
Poster presentations will be held 10:00 a.m. - 10:45 a.m. and 2:45 p.m. to 3:30 p.m. on Thursday, May 8, in the atrium of the Governor Calvert House. Posters will be set up for display by the conference staff. Presenters can drop off their posters at the registration desk by 5:00 p.m., Wednesday, May 7.

CONFERENCE PRESENTATIONS
Conference presentations and posters will be available online at [http://cps-hcss.org](http://cps-hcss.org).

HOTEL PARKING
Parking at the Historic Inns of Annapolis is by valet only. A reduced parking rate has been negotiated for overnight guests and daily conference attendees, both with in and out privileges. This reduced rate is $12/day. Daily parking for local government attendees is complimentary with approved government ID. Please visit the registration table each day to have your parking validated or the full fee may apply.

SURVEY
Please take a moment to respond to our short survey at: [http://cps-vo.org/group/hcss2014/survey](http://cps-vo.org/group/hcss2014/survey). Your valuable feedback will help us plan future conferences.
CONFERENCE ORGANIZATION

PROGRAM CO-CHAIRS

• Byron Cook, Microsoft Research & University College London
• John Hatcliff, Kansas State University

STEERING COMMITTEE

• Kathleen Fisher, DARPA
• John Launchbury, Galois, Inc.
• Brad Martin, National Security Agency
• Raymond Richards, Rockwell Collins
• Bill Scherlis, Carnegie Mellon University

ORGANIZER

• Katie Dey, Vanderbilt University
Tuesday, May 6  
Theme: Requirements and Specifications

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>Registration</td>
</tr>
</tbody>
</table>
| 0900  | **Keynote Presentation:** Going Native - Relying on Pidgins and Creoles to Construct High Confidence Software  
        Travis Breaux (Carnegie Mellon University) |
| 1000  | **High-Performance Regular Expression Processing for Cross-Domain Systems with High Assurance Requirements**  
        David Hardin (Rockwell Collins) |
| 1030  | Break                                                                |
| 1045  | **Programmable Hardware Support for Ubiquitous Micro-Policy Enforcement**  
        Benjamin Pierce (University of Pennsylvania) |
| 1115  | **Hardware-Based Tagging:** Building in Security from the Ground Up  
        Jim Alves-Foss (University of Idaho) |
| 1145  | **Formalization of x86 Machine-Code Programs with System Calls**  
        Warren Hunt (UT Austin) |
| 1215  | Lunch (on your own)                                                  |
| 1345  | **Generating Proof-Carrying Code for the UDP protocol**  
        Douglas Smith (Kestrel Institute) |
| 1415  | **SPARK 2014: Formal Program Verification For All**  
        Yannick Moy (AdaCore) |
| 1445  | **Going Large with Formal Methods on iFACTS**  
        Roderick Chapman (Altran UK) |
| 1515  | Break                                                                |
| 1530  | **Mining and Understanding Software Enclaves (MUSE)**  
        Suresh Jagannathan (DARPA) |
| 1600  | **Local Temporal Reasoning**  
        Eric Koskinen (New York University) |
| 1630  | **Through the Lens of Abstraction**  
        Aditya Thakur (University of Wisconsin) |
<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>Registration</td>
<td></td>
</tr>
<tr>
<td>0900</td>
<td>Designed-In Security Symposium</td>
<td><strong>Panel: Designed-In Security (DIS)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Moderator: Brad Martin (National Security Agency)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Panelists: Celia Merzbacher (SRC), Ron Perez (AMD), Bill Scherlis (CMU)</td>
</tr>
<tr>
<td>1000</td>
<td>A Building Code for Building Code</td>
<td>Carl Landwehr (Innovative Analytics and Training)</td>
</tr>
<tr>
<td>1045</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>1115</td>
<td>Panel: Moving forward with DIS and the Building Code</td>
<td>Moderator: Bill Scherlis (CMU)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Panelists: Byron Cook (Microsoft Research, UC London), Kathleen Fisher (DARPA), John Hatcliff (Kansas State University), John Launchbury (Galois, Inc.), Ray Richards (Rockwell Collins)</td>
</tr>
<tr>
<td>1215</td>
<td>Lunch (on your own)</td>
<td></td>
</tr>
<tr>
<td>1345</td>
<td>Mobility Session</td>
<td><strong>An Information Architecture Platform for Mobile, Secure, and Resilient Distributed Systems</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abhishek Dubey (Vanderbilt University)</td>
</tr>
<tr>
<td>1415</td>
<td>Multi-App Security Analysis: Looking for Android App Collusion</td>
<td>Rogan Creswick (Galois, Inc.)</td>
</tr>
<tr>
<td>1445</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>1515</td>
<td>Keynote Presentation: Mobile Computing - New Threats and New Solutions</td>
<td>Neal Ziring (National Security Agency)</td>
</tr>
<tr>
<td>1615</td>
<td>Triceratops: Privacy-protecting Mobile Apps</td>
<td>Edward Wu</td>
</tr>
<tr>
<td>1645</td>
<td>Text Analytics for Mobile App Security and Beyond</td>
<td>Tao Xie (UIUC)</td>
</tr>
<tr>
<td>1830</td>
<td>Conference Dinner (no host)</td>
<td>Chart House Annapolis</td>
</tr>
<tr>
<td>Time</td>
<td>Event</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0800</td>
<td>Registration</td>
<td></td>
</tr>
</tbody>
</table>
| 0900  | **Keynote Presentation:** DARPA's BRAIN Initiative  
Geoffrey Ling (DARPA) |
| 1000  | **Poster Session**                                                                         |
| 1045  | The Cyber-Physical Limits of Control  
Alex Wissner-Gross (Harvard University) |
| 1130  | Logical Foundations of Cyber-Physical Systems  
André Platzer |
| 1215  | Lunch (on your own)                                                                        |
| 1345  | Programming Languages for High-Assurance Autonomous Vehicles  
Lee Pike (Galois, Inc.) |
| 1415  | Toward Resilient Monitoring and Control of Distributed Cyber-Physical Systems  
Xenofon Koutsoukos (Vanderbilt University) |
| 1445  | **Poster Session**                                                                         |
| 1530  | Design and Implementation of Attack-Resistant Cyber-Physical Systems  
Miroslav Pajic (University of Pennsylvania) |
| 1600  | An Aircraft Electric Power System Domain-Specific Language for Reactive Control Protocols  
Huan Xu (University of Maryland) |
| 1630  | Conference End                                                                             |
Keynote:  
Going Native - Relying on Pidgins and Creoles to Construct High Confidence Software

Travis Breaux (Carnegie Mellon University)

Abstract:
Engineers developing high confidence systems must obtain a sound understanding of the domain phenomena, assumptions, and relevant invariants that underpin the space of acceptable solutions. As domain outsiders, these engineers must “go native” by engaging with domain experts to identify and discover the essential domain elements needed to properly exercise control over the environment in a predictable and reliable manner. Engineers and domain experts together rely on a variety of notations, methods, and tools to test their assumptions and to converge on a set of behaviors and properties that the system should embody, such as safety, security and privacy. This process is frequently riddled with ambiguity, inconsistency and conflict, which requires multiple design iterations, compromise and eventually commitment. In this keynote talk, we highlight the technical challenges to translating the informal, unstructured world into formal and semi-formal specifications needed to build and evaluate reliable, high confidence systems. These challenges include finding a domain-relevant semantics that is sufficiently expressive, supporting effective reasoning at the domain level, enabling a traceable connection between domain abstractions and implementation abstractions, developing mathematical metatheory to give confidence in this reasoning, and, in addition, assuring engineering realism and effective usability by engineers and evaluators.
Abstract:

Regular expressions are an important means for specifying search patterns in data. We recently responded to a customer requirement to provide regular expression support for a very high-assurance cross-domain system. The customer desired that complex regular expressions be used to specify the precise form of data payloads that would be allowed to cross domain boundaries. For such a critical system, it is essential that the processing of regular expressions is correct, and does not allow unintended data to transit from one domain to another. It is also important that regular expression matching is as efficient as possible; otherwise, the overhead of cross-domain filtering becomes too high to be useful.

We have addressed both the “designed-in security” and efficiency aspects of our customer’s requirements in the development of regular expression support for the Guardol domain-specific language for guards. We introduce a special Guardol expression, regex_match, which accepts a regular expression and a string, and produces a boolean output that indicates whether the string matches the regular expression. Every regex_match expression is translated into the application of a Deterministic Finite-State Automaton (DFA) to the given string. This translation is carried out in the “middle end” of the Guardol compilation process using the HOL4 theorem prover, and thus is highly trusted.

The DFA is generated using Brzozowski’s derivative technique, with modifications due to Owens, Reppy, and Turon. The DFA transition function, which takes a pair consisting of the current DFA state and the current character in the string, and returns the next state, is represented by a two-dimensional array. Thus matching a string is very quick, since there is one array indexing operation per character position in the string to be matched. Similarly, the final states are also represented in an array, so checking to see if the string is accepted is constant time. We have exercised the regular expression matcher on host platforms as well as the Rockwell Collins Turnstile guard, and performance is excellent.
In future work, we will provide support for proofs of Guardol properties that include regular expressions. The current Guardol code generator produces Ada code that is compiled and linked to form a guard executable; thus, to increase the trustworthiness of the Guardol backend, we will explore the elimination of the code generator/untrusted compiler part of the toolchain by compiling the functional form of Guardol directly to binary code using the verified CakeML compiler. We will also study direct synthesis of a hardware-only implementation of the regular expression matcher for very high-performance guard implementations.
Programmable Hardware Support for Ubiquitous Micro-Policy Enforcement
André DeHon and Benjamin Pierce
(University of Pennsylvania)

Abstract:
A host of security vulnerabilities arise from the violation of known, but in-practice unenforceable, safety and security policies, including high-level programming models and critical invariants of low-level programs. The onus is on every programmer to ensure that the entire system — every line of code — is free from vulnerabilities and exploitable errors, and remains that way as each feature is added and bug patched. Unlike safety-critical physical systems (cars, airplanes, chemical processing plants) that guard against life-threatening failures using redundant fail-safe mechanisms, present-day computer systems lack supervising safety interlock mechanisms to help prevent catastrophic failures that arise from violation of known safety and security policies.

This talk will demonstrate how a rich and valuable set of low-level micro-policies can be enforced at the hardware instruction-set level to provide such safety interlocks with modest performance impact. The enforcement of these micro-policies provides more secure and robust macro-scale behavior for our computer systems.

Specifically, we describe work developed in the context of the DARPA CRASH/SAFE project, and plans for follow-on efforts, to (i) introduce an architecture for ISA-level micro-policy enforcement; (ii) develop a linguistic framework for formally defining micro-policies; (iii) identify and implement a diverse collection of useful micro-policies; (iv) verify, through a combination of rigorous testing and formal proof, that combinations of hardware and software handlers correctly implement the desired policies and that the policies imply specific high-level safety and security properties; and (v) elaborate a microarchitecture to provide hardware support with low performance overhead and acceptable resource costs. We demonstrate how emerging hardware capabilities and advances in formal specification and verification can be combined to make it tractable to engineer systems with strong security and safety properties. We combine ideas from programming languages, verification, language-based security, hardware design, and microarchitectural optimization to provide a clean model for capturing policies and efficient mechanisms for implementing them.
Our proposed talk directly addresses the Requirements and Specifications topic of HCSS by emphasizing the formal specification and verification of our generic hardware and software mechanisms as well as specific micro-policies. Furthermore, it addresses the Designed-In Security theme, describing how to support ubiquitous policy enforcement at the lowest levels of the hardware/software stack. Our ubiquitous enforcement guarantees that key security behavior is enforced even when the code contains errors and vulnerabilities. Since the policies are separate from the code, their enforcement is guaranteed as the code evolves during its lifetime.

\[^1\text{http://www.crash-safe.org}\]
Abstract:

This talk provides an overview of hardware-based security tagging architectures (STA) and the use of these a building blocks for secure computing. This talk is based on a recently completed AFRL-funded study into the use of STAs, specifically on the benefits and limitations of this technology beyond the lab.

We will expand on the following:

STAs are known as promising mechanisms for enhancing the security of computer systems. A review of the literature demonstrates that the advantages of using security tagging schemes are quite clear. Security tagging was first designed and implemented to protect against a few low-level attacks, such as buffer overflow and format string attacks. Recently, security tagging schemes have been introduced that claim to support prevention of high-level attacks, which can include SQL injection and cross-site scripting. Tags are also implemented in some architecture to support memory access control. In this talk we will provide a brief overview of the existing approaches.

STAs can be used to detect type mismatches in memory access, control flow operations and machine code operations. The assignment of types to memory addresses and registers must be under the control of software. Therefore additional software, in the operating system or even at the middleware and application level, is needed to set the tags, and interpret errors generated by the hardware. This software will be more complex than traditional operating system memory protection software and will therefore require increased verification and validation. In this talk we will discuss the ramifications of this complexity to the design, implementations and verification of security solutions that utilize STAs.

STA hardware provides continual checks for type mismatches, relieving software of that burden, and providing greater confidence in the correct behavior of the system. However, care must be taken to not assume more functionality in the STA hardware than really exists, such as that found in some of the literature. Additional work is needed to understand the tradeoffs between STA supported security features and software-only based security features. This talk will provide examples of these misplaced assumptions and what additional work is needed.

In addition, in order to provide strong assurance of run-time enforceable security policies we contend that all executable hardware of the system (Direct
Memory Access (DMA) controllers, co-processors, network cards, etc.) will have to conform to STA principles, or will have to be isolated by STA hardware; a future area of research. This talk, given time, will discuss these concerns as well.
Abstract:
Most user-level programs use system calls to perform operations like I/O and memory management. Verification of these programs is required to assure their correct execution. We have extended our formal, executable model of the x86 instruction-set architecture (ISA), implemented in the ACL2 theorem-proving system, by adding support for system calls. We can now simulate and formally analyze user-level x86 machine-code programs that exhibit non-determinism caused by system calls and some x86 instructions, such as RDRAND, which is used in cryptographic applications.

Our model has two modes of operation: a logical mode that formalizes interaction with an external environment to support reasoning, and an execution mode that queries an underlying operating system to support simulation. We validate the execution mode with respect to both the logical mode and the real machine through extensive testing. This provides assurance that our model faithfully represents the semantics of an actual x86 machine. Using our model, we analyze x86 machine-code programs, which we usually obtain by running the GCC or Clang tool chain on C code. As a demonstration of the capabilities of our model, we mechanically verified a simple word-count (Linux-like “wc”) program that calculates the number of characters, lines, and words in an input stream.
Generating Proof-Carrying Code for the UDP protocol
Garrin Kimmel, Christoph Kreitz, Douglas R. Smith,
Eric W. Smith, Edwin Westbrook, Stephen Westfold
(Kestrel Institute)

Abstract:
The security of modern systems increasingly depends on securing their internal and external communications. We present a case study in the automated generation of proof-carrying code for a common communication protocol (UDP). The specification of the protocol is expressed in terms of trace predicates that capture both normal-case processing and the various errors that can arise and how to handle them. The disjunction of the trace predicates covers the specified behaviors of the protocol. The trace predicates are expressed in a mixed algebraic/coalgebraic style of specification in the Metaslang language of Kestrel’s Specware system. A derivation script is written manually that specifies a sequence of transformations from Specware’s library. Specware executes the derivation script automatically with the effect of incrementally refining the initial specification into idiomatic C code that runs in the Linux kernel.

A key part of this ongoing effort has been to instrument each transformation to emit a proof of the correctness of the refinement that it effects. To date, we can generate proofs for the first half of the transformation sequence. The proofs are expressed in the ISAR format of the Isabelle proof system, enabling Isabelle to automatically check them. In other words, our approach is to automatically and simultaneously generate both the code and proof. There is no need for post-hoc verification. The design information that is applied by a specification transformation is instantiated to produce both a refined specification and the logical justification of the refinement step.
SPARK 2014: Formal Program Verification For All

Yannick Moy (AdaCore)

Abstract:

SPARK is a programming language targeted at safety- and security-critical applications. SPARK builds on the strengths of the Ada programming language for creating highly reliable and long-lived software. SPARK restrictions ensure that the behavior of a SPARK program is unambiguously defined, and simple enough that formal verification tools can perform an automatic diagnosis of conformance between a program specification and its implementation. The SPARK language and toolset for formal verification have been applied over many years to on-board aircraft systems, control systems, cryptographic systems, and rail systems [4].

The latest version, SPARK 2014, builds on the new specification features added in Ada 2012 [1], so formal specifications are now expressed in the same language as programs, which also means that they are understood by the usual development tools and can be executed. Care was taken to match the formal proof semantics with the run-time assertion checking semantics (including possible run-time errors in formal specifications, and floating-point semantics). As a result, formal specifications can be tested and debugged like code, which goes a long way to making them accessible to all programmers.

Two other features of the new toolset contribute to this democratization of formal verification. The level of proof automation obtained by using state-of-the-art generators and provers of Verification Conditions makes fully automated proof within reach. User interaction is made practical and efficient through IDE integration, mechanisms for proof reuse, possible analysis of single components, and detailed feedback on failed proofs.

Yet, the most important enabler for industrial adoption is the integration in existing processes, for existing codebases. Formal verification can be combined with test of the executable produced with a special instrumentation of the compiler [2]. Going further in the direction of combining formal verification and other methods, we have proposed a framework for managing the assumptions upon which proof relies [3]. The toolset gives complete control to users over which components or subcomponents of a program are formally verified, which allows gradual adoption in brown fields.
The largest application of SPARK 2014 so far is a case study by the Space Systems division of Airbus Defence & Space, developed during three years along with the new language and tools. The success of this case study has prompted the decision to start a new one-year pilot project targeting the potential use of SPARK 2014 in the future Ariane 6. During the talk, I will use the Airbus Defense & Space case study to present the key features of the language and the tools, including a presentation of the actual user interaction inside an IDE, and I will finish with a focus on the combination of test and proof in relation to DO-178C.


Going Large with Formal Methods on iFACTS
Roderick Chapman (Altran UK)

Abstract:
The iFACTS system provides electronic flight-strip management, trajectory prediction and conflict detection for en-route air-traffic controllers in the UK. The development of iFACTS used “formal methods” in various contexts, including the functional system specification and the implementation, which is subject to various levels of static verification and proof. The formal specification is possibly one of the largest ever produced or reported, and the implementation runs to over 200kloc, with a team size that peaked at over 150. The system is in fully operational service, meaning 24 hours per day on all sectors.

Rather than focusing on the technical aspects of the iFACTS effort, this presentation will concentrate on the perceived usefulness, scalability, training needs, and metrics from the various “formal” aspects of the project, and the lessons that we’ve learned along the way.
Mining and Understanding Software Enclaves (MUSE)
Suresh Jagannathan (DARPA I20)

Abstract:
The Mining and Understanding Software Enclaves (MUSE) program at DARPA seeks to overcome the tremendous challenge of ensuring software correctness at scale. It proposes to do so through the creation of a community infrastructure that incorporates a continuously operational specification-mining engine applied to a large corpus of open-source software. The engine will leverage deep program analyses and foundational ideas underlying big data analytics to populate and refine a database containing inferences about the behavior and properties of programs inhabiting the corpus.

The overarching goal of the program is to effect a paradigm shift in the way software is conceived and maintained. This shift is realized by replacing the existing costly and laborious test/debug/validate cycle with ‘always on’ program analysis, mining, inspection and discovery over the constantly evolving corpus. Among other outcomes, we envision the creation of scalable automated mechanisms that can identify and repair program errors, as well as tools to efficiently synthesize custom programs from existing components based on a description of desired properties.
Local Temporal Reasoning

Eric Koskinen (New York University) and
Tachio Terauchi (Nagoya University)

Abstract:

Programming languages that use higher-order functionality (e.g. Java, C#, F#, Haskell, Ocaml, Perl, Python, Ruby) have become commonplace. Higher-order language features such as `map`, `grep`, Google’s Map/Reduce, are used widely and applauded for their simplicity and modularity.

Meanwhile, in the past couple of decades, algorithms and tools have emerged that have enabled automatic verification of some industrial software systems. Symbolic analysis techniques such as abstraction refinement and interpolation have given rise to interprocedural program analysis tools for safety such as SLAM and Blast, while termination argument refinement has lead to tools for liveness such as Terminator. Some of our recent work has led to tools and algorithms for verifying properties expressed in temporal logic: more elaborate specifications that combine safety and liveness.

These verification techniques have been mostly limited to imperative first-order software and cannot be applied to higher-order programming languages. In recent years, researchers have developed some techniques for verifying higher-order programs. Some showed how to verifying temporal properties of higher-order programs that are restricted to finite data. Others have also developed methods of verifying purely properties or purely termination of infinite-data higher-order programs.

Despite the efforts discussed above, at present there are no methods for verifying safety/liveness properties (i.e. temporal logic formulae) of programs written in higher-order languages.

**Compositional reasoning.** We present the first technique for verifying temporal logic properties of higher-order, infinite-data programs. The crux of our work is to decompose the problem, not only by dividing the program up into individual expressions via a type-and-effect system, but also, for every expression, to track the behavior of finite traces separate from the behavior of infinite traces. Our type rules permit verification oracles (including the type system itself) to reason about the conditional safety and liveness (i.e. temporal) behavior of program parts, and compose these facts together to prove the overall target property of the program. Moreover, we show that existing tools can be used as oracles to introduce liveness proofs into the type system’s effects. While it is a commonly held belief that type systems cannot be used for liveness properties, we show how
they can, nonetheless, be used to carry some liveness information and soundly combine reasoning about program parts together to prove overall safety and liveness for a wide variety of examples.

**Contributions.** To our knowledge, this work marks the first method for reasoning about temporal logic properties for higher-order programs that have infinite data. We believe our work provides the theoretical foundation toward several areas of practical significance. To this end, we have devised general rules so there are many instantiations and applications of them, including: (1) Instantiation to a wide variety of specification logics: able to support any logic that is closed under intersection, union, and composition (over finite and infinite traces) such as Büchi specifications. (2) Instantiation to arbitrary type environments. Often the type system alone is strong enough to derive safety properties. For example, when using refinement types in the absence of a termination oracle, our rules can be thought of as a novel extension to dependent types, where temporal behaviors are carried as effects. (3) Instantiation of oracles to any fragment of program expressions or any subset of the specification logic. (4) Instantiation to a modular reasoning system for temporal behaviors of first-order programs.

We have devised our methodology to be based on *local* reasoning, employing a type system. This stands in contrast to many existing verification works for higher-order programs that operate by extracting a transition system and then performing standard model checking techniques. Such existing techniques suffer from the inability to refine the abstraction during the verification process and, moreover, require input programs to be given in CPS form (or converted thereto), a further barrier to adoption in industry.
Through the Lens of Abstraction

Thomas Reps\textsuperscript{1,2} and Adita Thakur\textsuperscript{1}
\newline (\textsuperscript{1}University of Wisconsin, \textsuperscript{2}GrammaTech, Inc.)

Abstract:

This talk deals with the use of abstraction in two areas of automated reasoning: verification of programs, and decision procedures for logics.

Establishing that a program is correct is undecidable in general. Program-analysis and verification tools sidestep this tar-pit of undecidability by working on an abstraction of a program, which over-approximates the original program’s behavior. The theory underlying this approach is called abstract interpretation [1]. Abstract interpretation provides a way to obtain information about the possible states that a program reaches during execution, but without actually running the program on specific inputs. Instead, it explores the program’s behavior for all possible inputs, thereby accounting for all possible states that the program can reach.

Operationally, one can think of abstract interpretation as running the program “in the aggregate”. That is, rather than executing the program on ordinary states, the program is executed on abstract states, which are finite-sized descriptors that represent collections of states. For example, one can use abstract states that represent only the sign of a variable’s value: neg, zero, pos, or unknown. If the abstract state is $[a \rightarrow \text{neg}, b \rightarrow \text{neg}]$, the product “$a \times b$” would be performed as “neg*neg”, yielding pos. This approximation discards information about the specific values of $a$ and $b$: $[a \rightarrow \text{neg}, b \rightarrow \text{neg}]$ represents all states in which $a$ and $b$ hold negative integers.

However, there is a glitch: abstract interpretation has a well-deserved reputation of being a kind of “black art”, and consequently difficult to work with.

The first part of this talk will describe a fifteen-year quest to raise the level of automation in abstract interpretation, by presenting three different approaches to creating correct-by-construction analyzers:

1. The TVLA system [4] introduced a way to create abstractions of systems specified in first-order logic. Different analyses are defined using TVLA by varying the relation symbols of the logic, and, in particular, by varying which of the unary relations control how nodes are folded together. The specified set of relations determines the set of properties that will be tracked by the analyzer.

2. The TSL system [2] provides a framework for creating correct-by-
construction implementations of the state-transformation functions needed in tools that analyze machine code. From a single specification of the concrete semantics of a machine-code instruction set, TSL automatically generates state-transformation functions needed for static analysis, dynamic analysis, symbolic analysis, or any combination of the three.

3. Our recent work on symbolic methods for abstract interpretation [6] aims to bridge the gap between (i) the use of logic for specifying program semantics and program correctness, and (ii) abstract interpretation. Many of the issues can be reduced to the problem of symbolic abstraction:

Given a formula \( \varphi \) in some logic \( \mathcal{L} \), and an abstract domain \( \mathcal{A} \), find the most-precise descriptor \( a \) in \( \mathcal{A} \), that over-approximates the meaning of \( \varphi \).

The second part of the talk describes the use of abstraction in the design of decision procedures for logics. We start by explaining Stålmarck’s method, a decision procedure for propositional logic, using abstract-interpretation terminology [7]. In particular, we show how Stålmarck’s method is an instantiation of a generic framework parameterized by an abstract domain over Booleans. Furthermore, different instantiations of the framework lead to new decision procedures for propositional logic. Furthermore, this abstraction-based view allowed us to lift Stålmarck’s method from propositional logic to richer logics: to obtain a method for richer logics, instantiate the parameterized version of Stålmarck’s method with richer abstract domains [8]. We call such a decision-procedure design, which is parameterized by an abstract domain, a Satisfiability Modulo Abstraction (SMA) solver.

The talk will conclude by describing an SMA solver for separation logic [5]. Separation logic (SL) [3] is an expressive logic for reasoning about heap structures in programs, and provides a mechanism for concisely describing program states by explicitly localizing facts that hold in separate regions of the heap. SL is undecidable in general, but by using an abstract domain of shapes [4] we were able to design a semi-decision procedure for SL.

References


Panel: Designed-In Security
Moderator: Brad Martin (National Security Agency)
Panelists: Celia Merzbacher (SRC), Ron Perez (AMD),
and Bill Scherlis (Carnegie Mellon University)

Abstract:

Designed-In Security (DIS) is one of several research themes identified in the NITRD report, Trustworthy Cyberspace: Strategic Plan for the Federal Cybersecurity R&D Program. The focus of DIS is on enabling engineers to develop a system while simultaneously generating the assurance artifacts necessary to attest to the level of confidence in the system’s capabilities to withstand attack. The technical focus is on the development of assurance-focused engineering practices, languages, models, reasoning systems, and tools.

An initial workshop was held that brought together representatives from industry — primarily major software and hardware vendors — as well as government and research. The workshop discussion focused on understanding in a more concrete way the challenges facing industry, identifying significant emerging best practices, and developing a set of hard research problems most urgently in need of solution. The workshop attendees considered practices and challenges for both software and hardware, and also identified business-case issues associated with the adoption or potential adoption of emerging techniques.

The three panelists will present the results of the workshop, focusing on software, hardware, and business case. Future workshops are planned, focusing on industry sectors that, unlike the vendors, must rely on complex supply chains to develop and operate mission systems.
Abstract:
Cyberspace, though it has a physical reality of computers and communication channels, sensors and actuators, is in fact made real mostly by the programs that control those things. Today, systems of programs control most of our critical infrastructures. Metaphors are frequently used as a way to communicate to people what these programs are intended to do. Workers in cybersecurity have adopted many rich metaphors: Trojan Horse, virus, worm, firewall, and more. Difficulties arise when the metaphor blinds us to the underlying reality. The talk examines critically several common cybersecurity metaphors and proposes the adoption of a new (or at least underutilized) one, that of a building code for critical infrastructure software, as a means of putting what we have learned in forty years of system development experience into practice.
Moving forward with DIS and the Building Code

Moderator: Bill Scherlis (Carnegie Mellon University)
Panelists: Byron Cook (Microsoft Research & University College London), Kathleen Fisher (DARPA), John Hatcliff (Kansas State University), John Launchbury (Galois, Inc.), Ray Richards (Rockwell Collins)

Abstract:

There is a growing sense in the research community that we are, at long last, starting to make some initial technical steps towards the visionary goals identified in the two previous sessions -- on Designed-In Security and Building Codes. This includes progress in a diverse set of contributing HCSS disciplines, including models, languages, analysis and reasoning, lifecycle frameworks, architectural models, modeling of mission requirements and threats, human interaction and usability, tool design, etc.

This panel, comprised of the members of the HCSS Program Committee, will speculate on the goals of DIS and Building Code practices. What, for example, are some early indicators and incremental steps that might signal the best courses of action to pursue the vision? What is our sense, more broadly, of the technical opportunities and barriers to advancing the vision, and what research should be advanced? What are possible business and policy enablers or impediments?

The latter question, regarding the interplay of technology, business, and government requirements, was addressed as part of the “business case” portion of the DIS workshop, which took up ROI, risk factors, and potential benefits from research. Examples of issues include affordability and economic modeling, reputational and brand impacts, measurement challenges, government drivers, and intellectual property protection/loss.
An Information Architecture Platform for Mobile, Secure, and Resilient Distributed Systems

Abhishek Dubey¹, William R. Otte¹, Gabor Karsai¹, and Alessandro Coglio²
(¹Vanderbilt University, ²Kestrel Institute)

Abstract:
Networked platforms such as fractionated spacecraft that perform surveillance and swarms of UAVs that monitor the ground represent a new class of cyber-physical systems (CPS). These CPS are formed from distributed cyber and physical resources that provide a dynamic and mobile platform where the required resources, including available network bandwidth and computing nodes of the network can change at any time during the mission. Moreover, these systems host several mission-specific distributed applications, possibly supplied and used by organizations with different credentials, that must operate at different levels of criticality, simultaneously. The benefit of these platforms is their ability to be reconfigured for different missions with distributed applications that are developed, reused, and operated by different stakeholders.

A number of challenges exist. First, the platform must be able to guarantee resource and performance isolation for applications that might be sourced from different suppliers and need to operate at different criticality and security levels concurrently. As an example, consider a cluster of satellites that execute distributed applications. One application is a safety-critical cluster flight application (CFA) that controls the satellite’s flight and responds to emergency commands. Running concurrently with the CFA, image processing applications utilize the satellites’ sensors and consume much of the CPU resource. Neither interference through shared computing and network resources nor information leakage between these applications is permitted due to safety and security concerns.

Additionally, the system must provide strong resilience guarantees that ensure that the application deployment will be effectively managed and rearranged if either critical resources in the system, either physical or cyber, fail or if the performance of shared resources such as the network degrades substantially. Furthermore, the design tools used to develop applications of these systems must be able to produce verified systems with designed-in security such that resource capacity and security violations are detected and addressed early in the development cycle.
This talk describes a distributed Information Architecture Platform (IAP) that was developed and prototyped under the DARPA System F6 Program. It consists of two parts: (1) a design-time toolsuite for modeling, analysis, synthesis, integration, debugging, testing, and maintenance of application software built from reusable components; (2) a run-time software platform for deploying, managing, and operating application software on a network of embedded, mobile nodes connected via an ad-hoc network. The platform reduces the complexity and increases the robustness of software applications by providing reusable technological building blocks in the form of an operating system, middleware, and application management services. The talk will specifically focus on the key security concepts that have been designed into the platform and on how the platform guarantees strict resource isolation between applications.

The IAP implements a complete, end-to-end solution for software development: from modeling tools, which provide a design environment and support early analysis of software applications to identify violations of information flow security and resource usage, to a robust distributed runtime platform. The runtime platform includes layers that implement the high-level communication abstractions: synchronous and asynchronous interactions and an operating system. The operating system, prototyped as a set of extensions to the Linux kernel, implements all the critical low-level services to support resource sharing (including robust spatial and temporal partitioning), distributed process management (including process migration), secure (i.e. labeled and managed) information flows, and fault tolerance.

The other key feature of this platform is a novel kernel-level communication mechanism for providing secure information flows between processes called Secure Transport (ST). ST is a managed communications infrastructure that provides for datagram oriented exchange of messages between application tasks. ST restricts the transmission of datagrams according to both a communication topology according to a multi-domain multi-level Security (MLS) policy, both of which are configured for each task by a trusted system administration infrastructure.

Acknowledgments: This work was supported by the DARPA System F6 Program under contract NNA11AC08C. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of DARPA. The authors thank Olin Sibert of Oxford Systems and all the team members of our project for their invaluable input and contributions to this effort.

References:

[1] Tihamer Levendovszky, Abhishek Dubey, William Otte, Daniel Balasubramanian, Alessandro Coglio, Sandor Nyako, William Emfinger,


Multi-App Security Analysis:
Looking for Android App Collusion

Rogan Creswick (Galois, Inc.)

Abstract:
The Android security model was built from the ground up to combat potential attacks (or misuse) one app at a time. This model culminates in a user interface that asks for the user’s approval each time an app is installed. While the interfaces enables users to avoid applications that may violate their security policy (by using combinations of permissions) applications can freely communicate with each other to share their permissions, achieving capabilities through collusion that astute users would not have approved.

A holistic view of the app ecosystem is necessary to identify these potential collusions, but no existing tools provide such an interface. Galois has been working with DARPA to develop a system that enables security analysts to quickly examine collections of Android apps, focusing their analysis on sets of apps that may be sharing capabilities (either intentionally or inadvertently) that result in undesirable data flows and possibly exfiltration. The resulting tool, called FUSE, conducts a binary static analysis of individual Android apps to identify inputs and outputs. FUSE then produces an Extended Manifest that concretely defines the possible set of sources and sinks for a given app. An additional multi-app analysis connects compatible sources and sinks and presents users with an interactive interface where they can delve into the possible data flows for a given device.

This talk is a follow-on to Joe Hurd’s 2012 HCSS presentation about the vision of multi-app analysis in Android. Since then we have created a precise single-app static analysis, altered our approach to analysis based on analysis feedback, and implemented a deployable system for use when investigating new apps or designing a secure set of apps for a deployed device. In particular, the current approach does not assume that a security policy is known in advance. Instead, we allow the analyst to iteratively discover a policy during analysis—an approach that does not preclude the presence of a security policy a-priori, but also does not require a fixed security policy.
Abstract:
The ability to have secured computing and communications via mobile devices is now considered a mission imperative. From home users to government officials - from personal and business to Top Secret communication and information - everyone wants to work on smart phones, tablets, and other innovative mobile devices. This trend poses some unique threats, in addition to threats posed from traditional computing and communication methods. These unique threats partly arise from mobile device capabilities (e.g., location awareness, capabilities expose us to tracking threats) and partly arise from software deployment and use models (e.g., inter-app service models expose us to inter-app data theft threats). Fortunately the features offered by the new mobile platforms, and their relative freedom from legacy software burden, allow us to explore new means of addressing these threats. This talk explores how the principles of security architecture, software assurance, and trusted computing can be applied in the mobile environment to address emerging threats.
Triceratops: Privacy-protecting Mobile Apps
Edward X. Wu (University of Washington)

Abstract:
We propose to build a tool, Triceratops, for protecting user privacy in mobile applications. It allows any user to protect his or her personal information from malicious mobile apps, in ways that are not possible today. Triceratops takes a mobile app with a set of privacy policies as input, and generates a secured version of the app. The secured app dynamically and efficiently enforces the specified policies, guaranteeing privacy of its users.

More specifically, our tool provides (a) Finer grained control than current private data access models, and (b) Privacy guarantees, which ensure no false negatives (no missed alarms) regarding malicious leakage. These properties provide a more secure environment for users with smartphones. Furthermore, the following three properties make Triceratops practical: (1) Automated: The tool runs in a fully automated manner, with no manual code-inspection required. It also directly analyzes the compiled binary without requiring the access to source code. (2) Portable: Triceratops uses static code instrumentation to enforce the privacy policies by adding additional checks to the app. Thus it can be used to enforce privacy policies on any runtime environment without modifying the host OS. (3) Lightweight: Triceratops utilizes a novel technique that combines static optimization and runtime enforcement to achieve precise user privacy protection, while minimizing runtime overhead. We are excited that Triceratop’s enhanced privacy guarantee and ease of use have the potential to make secure mobile apps a reality.
Text Analytics for Mobile App Security and Beyond
Tao Xie (University of Illinois at Urbana-Champaign)

Abstract:
Mobile apps are accompanied by a rich amount of natural language text: app descriptions, app user reviews, update/release notes, etc. Such natural language text is essential in conveying important information about the apps (such as expected functionalities) and such information is not easily attainable from other structured information of the apps (such as app source or binary code, execution traces). Given the overwhelming amount of available natural language text, there is a high demand of text analytics including natural language processing (NLP) and text mining techniques to automatically analyze the natural language text to improve mobile app security. The history of applying NLP and text mining techniques to analyze software artifacts can date back to about a decade ago. Only till recently, text analytics for software artifacts such as mobile app artifacts has become an emerging research area in the security community. Our recent work has shown that automated analysis of natural language text can help improve mobile app security, and software security in general.

This talk highlights our WHYPER framework published in USENIX Security 2013 (http://web.engr.illinois.edu/~taoxie/publications/usenixsec13-whyper.pdf). In particular, while recent work has developed various techniques to determine what mobile apps do, no work has provided a technical approach to answer, what do users expect? Our WHYPER framework serves as the first step in addressing this challenge. WHYPER focuses on permissions for a given mobile app and examines whether the app description provides any indication for why the application needs a permission. WHYPER uses NLP techniques to identify sentences that describe the need for a given permission in an app description. The evaluation results demonstrate great promise in using NLP techniques to bridge the semantic gap between user expectations and application functionality, further aiding the risk assessment of mobile apps.

This talk also presents our Text2Policy approach (published in FSE 2012 http://web.engr.illinois.edu/~taoxie/publications/fse12-nlp.pdf), a text analytics approach for helping assure high application security during application development. In particular, Text2Policy extracts Access Control Policies (ACP) from natural language software documents such as requirements documents. ACPs specify which principals such as users have access to which resources. Ensuring the correctness and consistency of ACPs is crucial to prevent security
vulnerabilities. However, in practice, ACPs are commonly written in natural
language and buried in large documents such as requirements documents, not
amenable for automated techniques to check for correctness and consistency.
To address these issues, Text2Policy automatically extracts ACPs from natural
language software documents and resource-access information from natural
language scenario-based functional requirements.

This talk finally summarizes other major research work done in the area of
text analytics for software security, and outlines future research directions and
highlights research challenges in this emerging research area.
Keynote Presentation: DARPA’s BRAIN Initiative
Geoffrey Ling (DARPA)

Abstract:
In 2013, President Obama announced the Brain Research Through Advancing Innovative Neurotechnologies (BRAIN) Initiative, pledging $100M and charging the Defense Advanced Research Projects Agency (DARPA), the National Science Foundation, and the National Institutes of Health with accelerating the development and application of new technologies to show how the brain is organized, from individual cells to complex circuits, and how it behaves in real time. DARPA is continuing its existing neuroscience portfolio and launching new efforts that will leverage current research to improve understanding of brain function in order to alleviate the burden of illness and provide novel capabilities based on neurotechnology in the military and in everyday life. DARPA will also improve the ability of researchers to understand the brain through advancing tools for discovery and research via handling of data, novel imaging, and advanced analytics.
The Cyber-Physical Limits of Control
Alex Wissner-Gross (Harvard University)

Abstract:
As the speed of computer systems and their integration with the physical world have grown, the physical limits of control have become increasingly relevant for ensuring high confidence in software systems. In this talk, I will present exciting recent results on the physical limits in space and time for realizing optimal coordination and control in cyber-physical systems, and discuss their implications for the future of reasoning about control.
Abstract:
We study the logical foundations of cyber-physical systems (CPS), i.e. systems that combine cyber aspects such as communication and computer control with physical aspects such as movement in space. CPS applications abound. Ensuring their correct functioning, however, is a serious challenge. Scientists and engineers need analytic tools to understand and predict the behavior of their systems. That’s the key to designing smart and reliable control.

This talk identifies a mathematical model for CPS called multi-dynamical systems, i.e. systems characterized by combining multiple facets of dynamical systems, including discrete and continuous dynamics, but also uncertainty resolved by nondeterministic, stochastic, and adversarial dynamics. Multi-dynamical systems help us understand CPSs better, as being composed of multiple dynamical aspects, each of which is simpler than the full system. The family of differential dynamic logics surveyed in this talk exploits this compositionality in order to tame the complexity of CPS and enable their analysis.

In addition to providing a strong theoretical foundation for CPS, differential dynamic logics have also been instrumental in verifying many applications, including air traffic protocols for aircraft collision avoidance maneuvers, the European Train Control System protocol, several automotive systems, mobile robot navigation with the dynamic window algorithm, and a surgical robotic system for skull-base surgery.
Programming Languages for High-Assurance Autonomous Vehicles
Lee Pike (Galois, Inc.)

Abstract:
I will discuss how, in just 1.5 years, Galois built two new domain-specific languages (DSLs) and used them to develop a “hack-proof”, full-featured unpiloted air system. The secret to our increased productivity and assurance was building embedded DSLs, in which the DSLs are libraries in Haskell. We generate embedded C code that is guaranteed to be memory-safe from a relatively small specification.

The autopilot and more information is available at smaccmpilot.org.
Toward Resilient Monitoring and Control of Distributed Cyber-Physical Systems
Xenofon Koutsoukos (Vanderbilt University)

Abstract:
Security and resilience of Cyber-Physical Systems (CPS) is a significant problem due to wide deployment of commercial-off-the-self computing devices, network connectivity with the Internet, and the existence of organized, motivated attackers. Our goal is to complement security mechanisms with resilient algorithms for monitoring and control. Distributed monitoring and control algorithms provide redundancy as well as increased rate and fidelity of information. The agents share information in order to achieve specific group objectives, for example, consensus, formation control, synchronization, and distributed optimization.

This presentation describes a framework for design of resilient distributed protocols that are robust to adversarial attacks. The framework provides a theoretical characterization including conditions for stability and safety for arbitrary and dynamic network topologies. Resilient protocols for heterogeneous networks that include trusted agents are also presented. Finally, we consider cryptographic mechanisms that ensure data integrity and we present a game-theoretic model for decreasing the computational cost incurred in the network through probabilistic authentication of messages.
Design and Implementation of Attack-Resilient Cyber-Physical Systems
Miroslav Pajic, Nicola Bezzo, James Weimer, Oleg Sokolsky, George J. Pappas, Insup Lee
(University of Pennsylvania)

Abstract:
Modern embedded control architectures have moved from isolated systems to open architectures, such as new automotive systems with services that include remote diagnostics, code updates, and vehicle-to-vehicle communication. However, this increasing set of functionalities, network interoperability, and complexity of system design have also introduced security vulnerabilities that are easily exploitable, since the current embedded systems have not been built with security in mind. Furthermore, the tight interaction between information technology and physical world makes these systems vulnerable to malicious attacks beyond the standard cyber-attacks, and thus relying exclusively on cyber-security techniques is insufficient to provide guarantees on system performance. This is highlighted in cases when non-invasive sensor attacks occur (i.e., when the physical environment around a sensor is compromised to allow for injection of a malicious signal) as shown in recent attacks on Anti-lock Braking Systems and GPS sensors.

Consequently, there is a need to change the way we reason about control in Cyber-Physical Systems and to start designing attack-resilient control schemes and architectures capable of dealing with cyber-physical attacks on the environment of the controller (e.g., sensors, actuators, and communication media). To utilize knowledge of the system dynamics for attack detection and identification, we have to focus on a new set of problems such as attack-resilient state estimation, sensor and controller fusion under attacks, and detecting attacks in presence of noise and model uncertainty. In this talk, we present our recent efforts on the design of attack-resilient control schemes for cyber-physical systems. We focus on the problem of state estimation in presence of attacks, for systems with noise and modeling errors. When a state-based feedback controller uses the estimated states, we show that the attacker cannot destabilize the system by exploiting the difference between the model used for the state estimation and the real physical dynamics of the system. Furthermore, we describe how implementation issues such as jitter, latency, and synchronization errors can be mapped into parameters of the state estimation procedure that describe modeling errors, and provide a bound on the state-estimation error caused by the modeling errors. For a given control algorithm, such a robust method allows for the mapping of desired
control guarantees into real-time performance requirements from the underlying OS and networks, which facilitates reasoning about attack-resilience across different implementation layers. Finally, we illustrate the use of this approach on two real-world case studies by implementing attack-resilient cruise control on an unmanned ground vehicle and a state-of-the-art American Built Car.
An Aircraft Electric Power System Domain-Specific Language for Reactive Control Protocols

Huan Xu¹, Necmiye Ozay², Richard Murray³

(¹University of Maryland, ²University of Michigan, ³CalTech)

Abstract:

Domain-specific languages are languages adapted to a particular application or set of tasks. While general purpose languages (e.g., C or Java) may offer broader programming features, domain-specific languages (e.g., HTML or Verilog) provide more expressiveness and ease of use within a given domain [3]. Examples of languages used in the context of cyber-physical systems can be found in [1] and [2].

In aerospace, next-generation aircraft have moved away from purely mechanical and hydraulic subsystems, instead increasing reliance on electric power to supply subsystems, including flight-critical ones [4]. The growing complexity of electric power systems on aircraft, coupled with the need for safety, reliability, and autonomy, has increased the need to utilize formal methods and verification tools in order to properly analyze and design such large-scale systems. Previous work on embedded control software synthesis has been explored by Piterman and Pnueli [6]. Wongpiromsarn et al. [8] and Ozay et al. [5] have used the temporal logic planning toolbox TuLiP [9] to address the issue of creating correct-by-construction control protocols for aircraft management systems. Given a system topology, Xu et al. [10] show how text-based requirements can be converted to linear temporal logic (LTL), a formal specification language (see [7] for more details) to synthesize centralized and distributed controllers.

While the use of formal specification languages and correct-by-construction synthesis methods is beneficial in the area of controller design, unfamiliarity of formal methods amongst engineers may provide a challenge to widespread implementation of formal methods. We propose a domain-specific language that combines tools already in existence: visual programs for single-line diagrams, which engineers are familiar with, and primitives, which provide a more formal structure to specifications. The development of a domain-specific language, therefore, provides an easy interface between industry engineers knowledgeable in aircraft systems and the methods/tools used by computer scientists and software engineers.
This talk describes the use of a domain-specific language, and an accompanying software tool, in constructing correct-by-construction control protocols for aircraft electric power systems. Given a base topology, the language consists of a set of common “primitives,” which correspond to some set of standard specifications. The accompanying tool converts these primitives into formal specifications, which are used to synthesize control protocols. We then use TuLiP to synthesize centralized and distributed controllers. For systems with no time involved in the specifications, this tool also provides an option to output specifications into a SAT-solver compatible format, thus reducing the synthesis problem to a satisfiability problem. We provide the results of our synthesis procedure on a range of topologies.

References:

The poster presentations will be held 10:00 a.m. - 10:45 a.m. and 2:45 p.m. to 3:30 p.m. on Thursday, May 8, in the atrium of the Governor Calvert House.

**Perry Alexander, University of Kansas**
ArmoredSoftware: Trust in the Cloud

**Paul E. Black, NIST**
Toward Precise and Accurate Descriptions of Weaknesses

**Donald Goff, Cyber Pack Ventures**
Advancing Research in Security Science: Science of Security Effort

**Joseph Kiniry, Galois, Inc.**
Cryptol version 2: An Open Source Cryptol

**Kevin Krause, University of Idaho**
An Overview of an ACL2 Prototype C Integer Type Safety Verification Tool

**Rick Kuhn, NIST**

**Heather Lucas, National Security Agency**
Science of Security Virtual Organization (SoS VO)

**Kenneth Roe, The Johns Hopkins University**
Using Coq to Verify the DPLL algorithm

**Lucas Wagner, Rockwell Collins**
Specification and Analysis of Requirements (SpeAR) Tool

**Wei Yang, University of Illinois at Urbana-Champaign**
Towards Improving Mobile Application Security by Enhancing User Perceptions of Application Behaviors
ArmoredSoftware: Trust in the Cloud
Perry Alexander, Andy Gill, and Prasad Kulkarni (University of Kansas)

Abstract:
ArmoredSoftware is a new approach for establishing and maintaining trust among software systems operating in a cloud computing environment. Using remote attestation techniques, ArmoredSoftware components assess their environments and determine appropriate actions ranging from simply reporting issues to migration and mitigation. Each ArmoredSoftware component is packaged with lightweight measurement, attestation and appraisal capabilities that use TPM virtualization techniques to establish and maintain roots of trust. Appraisal capabilities request information from other collocated software components and the operational environment that respond by executing attestation protocols to gather and report evidence. Using techniques from trusted computing, attestation protocols also generate cryptographic signatures used to assess the trustworthiness of gathered evidence. In the appraiser, gathered evidence is used to assess the appraisal target while cryptographic signatures are used to assess trustworthiness of evidence.

This poster and accompanying demonstration represent the first public presentation of the ArmoredSoftware approach. We will outline the ArmoredSoftware architecture describing appraisal, attestation, and measurement. We will briefly present our evolving approach focusing on measurement and execution of attestation protocols. Our demonstration will show an end-to-end demonstration of a simple appraisal activity illustrating important ArmoredSoftware concepts.

*This work was sponsored in part by the United States, Department of Defense under contract H98230-13-C-0264.
Toward Precise and Accurate Descriptions of Weaknesses
Paul E. Black (NIST)

Abstract:
MITRE’s Common Weakness Enumeration (CWE) http://cwe.mitre.org/ is a list of several hundred classes of weakness that may be found in software. While it is a huge amount of progress over what was available a decade ago, there is still a lot of work to do. We propose some directions to significantly improve CWEs. These directions come from semantic templates, software fault patterns, and other work.

To motivate our proposal, we give examples of some ambiguities, gaps, and problems that we found while checking the SATE V Ockham Sound Analysis Criteria http://samate.nist.gov/SATE5OckhamCriteria.html Even “simple” CWEs, such as uninitialized variable, don’t correspond well to the warning classes that static analysis tools produce. For instance does CWE-457: Use of Uninitialized Variable cover the case when just one field of a structure is not initialized before it is used? Or does that fall under a far-more-general CWEs, like CWE-824: Access of Uninitialized Pointer, CWE-665: Improper Initialization, CWE-824: Access of Uninitialized Pointer, or CWE-908: Use of Uninitialized Resource?
Advancing Research in Security Science: Science of Security Effort

Donald Goff (Cyber Pack Ventures, Inc.)

Abstract:
The Science of Security project entails support to inventory, catalog and disseminate research efforts currently underway at the nation’s universities and to determine how that research relates to the development and maturation of the science of cybersecurity. The purpose of the project is to identify promising fields and determine how they relate to the creation of a broad scientific body of knowledge and the development of basic precepts in this new field. This poster demonstration will show the project’s work to date, including opportunities available to researchers, current bibliography, and other items of interest collated for periodic highlights on the Science of Security Virtual Organization web site.
CONFERENCE POSTERS

Cryptol version 2: An Open Source Cryptol
Joseph Kiniry (Galois, Inc.)

Abstract:
HCSs participants, in the main, know about Galois’ Cryptol language and system and its capabilities. In short, Cryptol is a domain-specific language for programming, executing, testing, and formally reasoning about streams of bits. Cryptol particularly excels at specifying and reasoning about cryptographic algorithms.

Galois has decided to “reboot” Cryptol and create, from the ground up, a new Open Source Cryptol release: Cryptol version 2.

Cryptol version 2 is the first Open Source release of the Cryptol system. Its purpose is to make rigorous applied cryptography available to all. Moreover, we advocate that Cryptol should be used as the foundation for the specification and verification of past and future cryptographic algorithms standardized via NIST competitions and other similar activities.

The Cryptol system provides: (1) a REPL for experimentation, (2) a parser and typechecker for Cryptol programs, (3) an interpreter for executing Cryptol programs, (4) a validation tool for gaining confidence in the correctness of Cryptol programs via automatic randomized testing, and (5) a verification backend for formally verifying properties of Cryptol programs through the use of SMT solvers.
An Overview of an ACL2 Prototype C Integer Type Safety Verification Tool
Kevin Krause (University of Idaho)

Abstract:
The preeminent status that the C programming language continues to enjoy comes at the cost of a lack of memory bounds checking. For example, it is widely known that C is not type safe because its integer types are highly susceptible to entering an error condition. A single integer error might cause a system to exhibit unexpected behaviors. Moreover, the error can leave an entire system vulnerable to failure or even to a hostile takeover. If such an error manifests in a critical system, the consequences could be devastating. Such errors occur both in mobile systems as well as control systems. Analysis of the code in these systems is an essential building block for ensuring the correctness of security solutions.

Several solutions, such as safe coding guidelines, libraries of safe integer operations, and type safe subsets of C, have been proposed to ameliorate the likelihood integer errors. Except for the better coding practices, the solutions may not be practical for every situation. Libraries of safe operations require added operational overhead that could exceed the limits low powered or other real time systems; and are therefore ignored in control system and mobile systems. Likewise, type safe subsets ties the hands of programmers by forcing them to use cumbersome procedures when the use of mixed data types are required.

Because the onus of writing type safe C programs remains squarely with the programmers, their code should be rigorously tested throughout the development process. Several tests have been developed to achieve this end. Some are designed to verify type safety. However, no two of these test are the same in their methods, logic, and reporting. As a result, wide gaps in both accuracy and coverage remain in the realm of type safety verification. We are developing a prototype C integer type safety verification tool with a goal of closing that gap.

This poster presents an overview of the steps we took to develop our tool. That includes the identification of the problem space and a formalization of the static typing semantics of C expressions and statements based on the typing constraints on each program construct. From those two steps, the operational semantics of our tool were developed. The algorithms are supported by annotated lookup tables that gives our ability to track state wise changes. For example, entities in the data object table contain unique identification, typing, value, and/or any error-warning message information. Finally, there is the customizable output of our tool that can be proven to be correct.

Jon Hagar¹, Rick Kuhn², Raghu Kacker³, Tom Wissink³

(¹Grand Software Testing & Lockheed Martin
²NIST, ³Lockheed Martin)

Abstract:

This poster gives an overview of the experience of eight pilot projects, over two years, applying combinatorial testing in Lockheed Martin (LM), one of the world’s largest aerospace firms. Lockheed Martin and NIST developed a Co-operative Research and Development Agreement (CRADA) to evaluate effectiveness and areas of suitable application for combinatorial testing in a real-world industrial setting with complex software requirements. (One of the ways in which NIST conducts joint research with US industry is through CRADAs, which allow federal laboratories to work with US industry and provide flexibility in structuring projects, intellectual property rights, and in protecting industry proprietary information and research results.)

Objectives in the pilot project evaluation included:

1) Investigate applicability of CT in a variety of application areas, including system, software, and hardware testing;
2) Determine effectiveness of CT for improving fault detection; and
3) Study potential for reducing test cost or overall lifecycle cost by finding errors earlier in the process.

Software Tools

The primary tool for most projects was ACTS. Additional tools with complementary capabilities were also used in the pilot projects.

- NIST & U. of Texas Arlington: ACTS
- Air Academy Associates: SPC XL, DOE KISS, DOE PRO XL, DFSS MASTER
- Phadke & Associates: rdExpert
- Hexawise: Hexawise tool
Application Areas
Eight pilot projects were identified, with the goal of using the new methods in areas with diverse testing needs:

- Flight Vehicle Mission Effectiveness (ME) – comparing CT with tests generated from a statistical analysis tool
- Flight Vehicle engine failure modes – compared CT tests with existing tests developed using previous practice
- Flight Vehicle engine upgrade – tests including combinations of flight mode factors; comparison with existing tests
- F-16 Ventral Fin Redesign Flight Test Program – application to problem analysis (system-level evaluation rather than software testing)
- Electronic Warfare (EW) system testing – evaluating and extending existing tests
- Navigation Accuracy, EW performance, Sensor information, and Radar detection – generating test cases for subsystems
- Electromagnetic Effects (EMI) Engineering compared CT tests with existing tests developed using previous practice
- Digital System Command testing – testing file functions with multiple parameters

Results and Evaluation

While results varied across the different pilot projects, overall it was estimated that CT would save roughly 20% of testing cost, with 20% - 50% improved test coverage. In some cases, significant, previously undetected bugs were discovered. Additional findings included:

Positive results - Demonstrated the ability to reduce test cost in a variety of areas; teams found many tools practical

Mixed results – Reluctance of many engineers to adopting new methods; some teams did not identify significant improvements

Lessons learned – Most critical factors affecting adoption: availability of education and training for the new method; clear demonstration of value.

Recommendations

- Develop and improve education and training materials
- Incorporate combinatorial methods into DoD guidance and industry standards; best practices
CONFERENCE POSTERS

• Expand internal company guidance – developing a community of practice
• Greater availability of tools to support combinatorial testing – improved usability; matching tool to problem
• Modify approaches to using combinatorial testing – integrating combinatorial testing with other test practices; ability to adopt CT partially or gradually

Certain products may be identified in this document, but such identification does not imply recommendation by the US National Institute of Standards and Technology or other agencies of the US Government, nor that the products identified are necessarily the best available for the purpose.
Science of Security Virtual Organization (SoS VO)
Heather Lucas (National Security Agency)

Abstract:
While certain subfields of security have a strong scientific basis (e.g., cryptography, formal methods), there is no comprehensive scientific basis for constructing systems that are trustworthy by design. This lack of a disciplined and rigorous scientific basis profoundly limits our ability to design, deploy, and trust most large-scale and cyber-physical systems (CPS). A foundational science of security is needed now. To this end, the Science of Security Group will provide a virtual organization for community awareness, collaboration, and information all directed toward the maturing of the scientific basis for security.

The Science of Security Virtual Organization (SoS VO) enables those interested in cybersecurity science to survey current research; stay current on news in the field; find out about events related to cybersecurity science; collaborate with others using chat, video conferencing, and forums; share work by uploading documents and creating wikis; and access educational resources contributed by members.
Using Coq to Verify the DPLL algorithm

Kenneth Roe (The Johns Hopkins University)

Abstract:
Most large software systems have complex data structures with complex invariants. Many bugs can be traced to code that does not maintain these invariants. As an example, the Heartbleed bug can be traced to an attacker sending a packet that violates an invariant that is assumed to be true in the OpenSSL code. Often these invariants are not well documented. However, maintaining them is necessary for correct operation of the software.

In order to develop techniques for verifying complex invariants, we are in the process of developing a verification of a simplified DPLL algorithm. We have a C implementation of this algorithm which is about 200 lines of code. The code implements the two watch variable algorithm for efficient unit propagation. Our proof verifies that the code properly maintains the watch variable invariant. This invariant is described using separation logic and consists of 100 lines of Coq code.

The verification uses our PEDANTIC framework which is designed for verifying the correctness of C-like programs using the Coq theorem prover. PEDANTIC is designed to prove invariants over complex dynamic data structures such as inter-referencing trees and linked lists. The language for the invariants is based on separation logic. The PEDANTIC tactic library has been constructed to allow program verifications to be done with reasonably compact proofs.

While working on the DPLL example, we have found existing Coq IDE tools to not be well suited for some of the tasks needed for a complex verification task. For example, if while verifying an entailment, we find an error in the invariant, then all theorems that have been completed need to be modified and their scripts revised. It would be nice to have an intelligent replay that can guess at changes that need to be made to the scripts. We are in the early stages of developing an IDE based on Python. In addition to replay, the IDE will present a scripting language in which programs can be written to automate some editing tasks. We will an overview of this IDE in our poster.

For more information on this research, check out the website at http://www.cs.jhu.edu/~roe.
Specification and Analysis of Requirements (SpeAR) Tool
Kerianne H. Gross¹, Lucas G. Wagner², Aaron W. Fifarek³
(¹Air Force Research Laboratory, ²Rockwell Collins, ³LinQuest Corporation)

Abstract:
Traditional safety-critical software development methods focus on the sequential activities of specification, design, implementation, and verification. The inherent weakness in this approach is that any specification or design errors found in verification require rework through all steps of the process. This is compounded by the growing complexity of software, often resulting in substantial software development overruns in safety-critical applications. In this paper we introduce a formalized requirements development framework named Specification and Analysis of Requirements (SpeAR) that is designed to help users develop more precise specifications which are formally constructed and enable analysis early in system development. SpeAR provides a set of commonly used patterns as a front-end for behavioral specification and a suite of analyses on the back-end to ensure the resulting specification is logically consistent. This paper will discuss the features of SpeAR and briefly discuss one industrial effort in which it was used to specify and analyze requirements.

References


Towards Improving Mobile Application Security by Enhancing User Perceptions of Application Behaviors

Xusheng Xiao¹ Rahul Pandita¹ Wei Yang² William Enck¹ Tao Xie²
¹North Carolina State University, ²University of Illinois at Urbana-Champaign

Abstract:

Application markets such as Apple’s AppStore and Google’s Play store provide easy mechanisms for developers to distribute their applications. Unfortunately, these markets also provide easy mechanisms for developers with ulterior intent to distribute malware. However, classifying an application as malicious, privacy infringing, or benign is an ongoing challenge in the application markets. Existing formal analysis tool [1–5,7,9,10](directed towards detecting malicious applications) do not make a distinction between user expected application behavior and the unexpected application behavior, and may potentially report all security and privacy sensitive operations as malicious. This poster presents our works on bridging the semantic gap between user perceptions of the application behavior and the actual application behavior.

In our first project [8], we provide a user-aware privacy control approach that: 1) notifies users of potential information leak via presenting information flows that show what private data types are flowing to what output channels, and 2) allows users to perform inspection of the outgoing information at runtime. However, some information flows can flow to output channels where users cannot perform runtime inspection such as network socket (referred to as escaping flows), and may tamper with the information before the information is presented to users for inspection (referred to as tampering flows). To differentiate such information flows with other information flows where users can inspect un-tampered information, our approach further provides tamper analysis that tracks whether information is tampered with before the information flows to output channels, and identifies escaping flows and tampering flows for users to inspect.

In our second project [6], we developed an approach to automatically identify sentences that indicate the uses of security permissions in the application descriptions. Specifically, in this work, we specialized Natural Language Processing (NLP) techniques using the domain-specific models inferred from API documents to distinguish such sentences from the others. These domain-specific models describe various actions that perform on the resources protected by permissions, representing common uses of permissions. Our evaluation
results on about 600 application descriptions show great promise in using NLP techniques to bridge the semantic gap of user expectations to aid the risk assessment of mobile applications.

References


The conference dinner will be held at the Chart House restaurant on Wednesday, May 7, 2014 at 6:30 p.m. Within walking distance of historic downtown Annapolis, Chart House offers fantastic waterfront views of City Dock, the state capital, and the U.S. Naval Academy. Located in the Eastport section of Annapolis, the restaurant has ample parking and is accessible by water taxi. For persons attending the dinner, tickets can be purchased (cash or check) at the registration desk.

300 Second Street
Annapolis, MD 21403
Phone: 410.268.7166

Directions from the Governor Calvert House

Driving
- Head northwest toward Maryland Ave
- Exit the traffic circle onto School St
- Turn right onto Church Circle
- Turn right onto Duke of Gloucester St
- Slight right onto Compromise St
- Continue onto 6th St
- Turn left onto Severn Ave
- Turn left onto 2nd St
- Destination will be on the left

Walking
- Head south toward East St
- Exit the traffic circle onto Francis St
- Turn left onto Main St
- At the traffic circle, continue straight to stay on Main St
- Continue onto Compromise St
- Continue onto 6th St
- Turn left onto Severn Ave
- Turn left onto 2nd St
- Destination will be on the left
LOCAL RESTAURANTS

Cantler’s Riverside Inn – 458 Forest Beach Rd., Annapolis, MD 21401
“The crabs are top notch, the view is without parallel.”

Davis’ Pub – 400 Chester Avenue, Annapolis, MD 21403
“Featured on Diners, Drive-Ins and Dives. The crab pretzel is to die for.”

Dock Street Bar & Grill – 136 Dock Street, Annapolis, MD 21401
“Dock Street features Chesapeake Bay cuisine served daily until 1 a.m.”

Harry Browne’s – 66 State Circle, Annapolis, MD 21401
“A captivating historic restaurant/lounge. Lavish lunches, divine dinners and sumptuous Sunday brunch.”

Joss Café & Sushi Bar – 1959 Main Street, Annapolis, MD 21401
“Voted ‘Best Sushi Restaurant’ in Annapolis for 8 years running by the readers of What’s Up? Magazine.”

Lemongrass – 167 West St., Annapolis, MD 21401
“Lemongrass serves guests fresh, authentic Thai Cuisine in a warm contemporary environment.

O’Brien’s Oyster Bar – 113 Main St., Annapolis, MD 21401
“Located at the foot of Main Street, offering imaginative seafood dishes and nouveau American cuisine. Dancing and live entertainment nightly.”

Paladar – 1905 Town Center Blvd., Annapolis, MD 21401
“This Latin kitchen and rum bar offers a killer happy hour menu of $5 food & drink items.” (Katie recommends the fish tacos!)

Rams Head Tavern – 33 West St., Annapolis, MD 21401
“An Annapolis landmark since 1989! Enjoy great food, cold brews, and a friendly atmosphere in an English style pub or cozy dining room.”

Ristorante Piccola Roma – 200 Main Street, Annapolis, MD 21403
“Featuring classic Italian favorites, as well as innovative dishes, this restaurant offers a select wine list to complement both the lunch and dinner menus.

Vin 909 – 909 Bay Ridge Ave., Annapolis, MD 21403
“Great food. Great wine. Excellent service. Reasonable prices.”