THE FIFTEENTH ANNUAL
HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE
Annapolis, MD | May 5-7, 2015
http://cps-hcss.org
The Fifteenth Annual

High Confidence Software and Systems Conference

Annapolis, Maryland, USA | May 5-7, 2015
http://cps-hcss.org

Program Co-Chairs
Kathleen Fisher, Tufts University
John Hatcliff, Kansas State University

Steering Committee
Byron Cook, University College London
John Launchbury, DARPA
Stephen Magill, Galois, Inc.
Brad Martin, National Security Agency
William Scherlis, Carnegie Mellon University

Conference Sponsor
NITRD HCSS Coordinating Group
The Co-Chairs, NITRD HCSS Coordinating Group, and steering committee are pleased to welcome you to the 15th 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 fifteen-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 Proof Engineering, Sustainable Integrity and Privacy. 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 2015 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.

Kathleen Fisher
Tufts University

Byron Cook
University College London

Brad Martin
National Security Agency

William Scherlis
Carnegie Mellon University

John Hatcliff
Kansas State University

John Launchbury
DARPA

Stephen Magill
Galois, Inc.
Welcome Message ....................................................................................2
Table of Contents ....................................................................................3
General Information .................................................................................4
Conference Organization ........................................................................5
Program Agenda .....................................................................................6
Conference Presentations .......................................................................9
Poster Presentations .............................................................................43
Conference Dinner ...............................................................................61
Local Restaurants ...............................................................................62
Notes ....................................................................................................64
GENERAL INFORMATION

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

- 11:00 a.m. to 6:00 p.m. Sunday
- 7:00 a.m. to 5:30 p.m. Monday
- 8:00 a.m. to 4:00 p.m. Tuesday
- 8:00 a.m. to 4:30 p.m. Wednesday
- 8:00 a.m. to 5:00 p.m. 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
The poster session will be held between 10:30 a.m. - 11:15 a.m. and 3:00 p.m. – 3:45 p.m. on Thursday, May 7, 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 9:00 a.m. on Thursday, May 7.

CONFERENCE PRESENTATIONS
Conference presentations and posters will be available online at 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 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. Government attendees should 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/hcss2015/survey. Your valuable feedback will help us plan future conferences.
CONFERENCE ORGANIZATION

PROGRAM CO-CHAIRS
• Kathleen Fisher, Tufts University
• John Hatcliff, Kansas State University

STEERING COMMITTEE
• Byron Cook, University College London
• John Launchbury, DARPA
• Stephen Magill, Galois, Inc.
• Brad Martin, National Security Agency
• William Scherlis, Carnegie Mellon University

ORGANIZERS
• Katie Dey, Vanderbilt University
• Anne Dyson, Innovative Analytics & Training
## PROGRAM AGENDA

### Tuesday, May 5

**Theme: Proof Engineering**

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>Breakfast</td>
</tr>
</tbody>
</table>
| 0900  | **Keynote Presentation:** Proof Engineering: The Soft Side of Hard Proof  
Gerwin Klein (NICTA) |
| 1000  | **Cerberus: Towards an Executable Semantics for Sequential and Concurrent C11**  
Kayvan Memarian (University of Cambridge) |
| 1030  | Refreshments                                                                                     |
| 1100  | **Verifiable C: Proving Functional Correctness of C Programs in Coq, e.g. SHA-256 and HMAC**  
Andrew Appel (Princeton University) |
| 1130  | **Deep Specifications and Certified Abstraction Layers**  
Ronghui Gu (Yale University) |
| 1200  | Lunch (on your own)                                                                             |
| 1330  | **Qualification of Formal Methods Tools**  
Darren Cofer (Rockwell Collins) |
| 1400  | **Issues, Challenges, and Opportunities in the Qualification of Formal Methods Tools**  
Cesare Tinelli (University of Iowa) |
| 1430  | Refreshments                                                                                     |
| 1500  | **Multi-Language and Multi-Prover Verification with SAWScript**  
Aaron Tomb (Galois, Inc.) |
| 1530  | **CodeHawk: Sound Static Analysis for Proving the Absence of Memory Related Software Vulnerabilities**  
Douglas Smith (Kestrel Technology) |
| 1600  | Adjourn for the Day                                                                               |
# PROGRAM AGENDA

## Wednesday, May 6
*Theme: Sustainable Integrity*

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>Breakfast</td>
</tr>
</tbody>
</table>
| 0900  | Keynote Presentation:  
       Detecting Malice in Commodity Software  
       Tim Fraser (DARPA)                              |
| 1000  | Remote Attestation for Cloud-Based Systems  
       Perry Alexander (University of Kansas)          |
| 1030  | Refreshments                                                                      |
| 1100  | Software Defenses Inspired by Biodiversity  
       Michael Franz (University of California, Irvine) |
| 1130  | Not-quite-so-broken TLS: Lessons in Re-engineering a  
       Security Protocol Specification and Implementation  
       David Kaloper Meršinjak (University of Cambridge) |
| 1200  | Lunch (on your own)                                                               |
| 1330  | Rigorous Architectural Modelling for Production Multiprocessors  
       Shaked Flur, Kathryn Gray, and Christopher Pulte  
       (University of Cambridge)                           |
| 1400  | A Formal Specification of x86 Memory Management  
       Shilpi Goel and Warren Hunt (UT Austin)            |
| 1430  | Language-based Hardware Verification with ReWire:  
       Just Say No! to Semantic Archaeology  
       William Harrison (University of Missouri)          |
| 1500  | Refreshments                                                                      |
| 1530  | Achieving High Speed and High Assurance in a  
       Hardware-Based Cross-Domain System using Guardol  
       David Hardin (Rockwell Collins)                  |
| 1600  | Bringing Hardware Hacking to Life  
       Colin O’Flynn (Dalhousie University)              |
<p>| 1630  | Adjourn for the Day                                                               |</p>
<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>Breakfast</td>
</tr>
<tr>
<td>0900</td>
<td><strong>Keynote Presentation:</strong> Building Privacy-Aware Computing Systems: An Overview of Current Capabilities and Technical Challenges</td>
</tr>
<tr>
<td></td>
<td>Shantanu Rane (PARC)</td>
</tr>
<tr>
<td>1000</td>
<td>Reconciling Provable Security and Practical Cryptography: A Programming Language Perspective</td>
</tr>
<tr>
<td></td>
<td>Gilles Barth (IMDEA Software Institute)</td>
</tr>
<tr>
<td>1030</td>
<td><strong>Poster Session &amp; Refreshments</strong></td>
</tr>
<tr>
<td>1115</td>
<td>A Cut Principle for Information Flow</td>
</tr>
<tr>
<td></td>
<td>Joshua Guttman (The MITRE Corporation and WPI)</td>
</tr>
<tr>
<td>1145</td>
<td>Models and Games for Quantifying Vulnerability of Secret Information</td>
</tr>
<tr>
<td></td>
<td>Piotr Mardziel (University of Maryland)</td>
</tr>
<tr>
<td>1215</td>
<td>Lunch (on your own)</td>
</tr>
<tr>
<td>1345</td>
<td><strong>NSA Civil Liberties &amp; Privacy:</strong> Bridging the Art and Science of Privacy</td>
</tr>
<tr>
<td></td>
<td>Rebecca Richards (National Security Agency)</td>
</tr>
<tr>
<td>1430</td>
<td>High Assurance Cryptography Synthesis with Cryptol</td>
</tr>
<tr>
<td></td>
<td>Joseph Kiniry (Galois, Inc.)</td>
</tr>
<tr>
<td>1500</td>
<td><strong>Poster Session &amp; Refreshments</strong></td>
</tr>
<tr>
<td>1545</td>
<td>Privacy through Accountability</td>
</tr>
<tr>
<td></td>
<td>Anupam Datta (Carnegie Mellon University)</td>
</tr>
<tr>
<td>1630</td>
<td>Private Disclosure of Information</td>
</tr>
<tr>
<td></td>
<td>Daniel Aranki (UC Berkeley)</td>
</tr>
<tr>
<td>1700</td>
<td>Conference Adjourned</td>
</tr>
</tbody>
</table>
Proof Engineering: The Soft Side of Hard Proof
Gerwin Klein (NICTA)

Abstract:
We do formal machine-checked proof because it produces mathematical theorems and hard guarantees. But there is also a softer side: the process by which we arrive at the final proof. Especially in larger-scale formal proofs such as in the Odd-Order theorem or in program verification such as the correctness proof of the seL4 microkernel, issues of proof engineering become important factors in determining success or failure. This talk will give an overview of proof engineering problems in large-scale proofs, and present some of the solutions employed in the seL4 verification. A particular open problem in proof engineering is effort prediction. I will give an overview of recent preliminary statistical results on the correlation of proof size and effort and of specification size and proof size.
Cerberus: Towards an Executable Semantics for Sequential and Concurrent C11

Kayvan Memarian, Kyndylan Nienhuis, Justus Matthiesen, *James Lingard, Peter Sewell
(University of Cambridge and *Microsoft Research Cambridge)

Abstract:

C remains central to our computing infrastructure, but still lacks a clear and complete semantics. Programmers lack tools to explore the range of behaviours they should expect; compiler development lacks test oracles; and formal verification and analysis must make (explicitly or implicitly) many choices about the specific C they target.

We describe Cerberus, a semantics for a substantial fragment of C11. Its thread-local semantics is factored via an elaboration into a simpler Core language, to make it conceptually and mathematically tractable. This is integrated with an operational model for C11 concurrency, with a mechanised proof of equivalence to the axiomatic C11 model of Batty et al. The front-end includes a parser that closely follows the C11 standard grammar and a typechecker. Cerberus is executable, to explore all behaviours or single paths of test programs, and it supports proof, as shown by a preliminary experiment in translation validation for the front-end of Clang, for very simple programs. This is a step towards a clear, consistent, and unambiguous semantics for C.
Verifiable C: Proving Functional Correctness of C Programs in Coq, e.g. SHA-256 and HMAC
Andrew Appel (Princeton University)

Abstract:

*Verifiable C* is a program logic for C—a higher-order impredicative concurrent separation logic. As part of the Verified Software Toolchain, it comes with automated+interactive tactical proof tools for applying the logic to C programs. Verifiable C is proved sound w.r.t. the operational semantics of C; this proof connects to Leroy et al.’s correctness proof for the CompCert optimizing C compiler. All these proofs (including user proofs) are machine-checked in the Coq proof assistant. Composed together inside Coq, they guarantee that the functional-correctness properties you prove about the source program, are obeyed by the assembly-language (compiled) program.

Verifiable C is generally applicable to C programs; we demonstrate it on SHA-256 and HMAC lightly adapted from the OpenSSL implementations. We prove SHA and HMAC correct with respect to the FIPS 180 and FIPS 198 standards (respectively), and connect to Coq proofs of the high-level cryptographic properties of HMAC.

I’ll use these examples to show the architecture of a functional-correctness specification: (1) pure functional spec, (2) proving user-relevant properties of functional specs, (3) API spec, which relates functional spec to the Application Programmer Interface of a C program, (4) proof that a program satisfies the API spec. HMAC’s user-relevant property is that it implements a PRF (pseudorandom function), subject to certain standard (and fully formalized) assumptions about SHA-2.
Deep Specifications and Certified Abstraction Layers
Ronghui Gu (Yale University)

Abstract:
Modern computer systems consist of a multitude of abstraction layers (e.g., OS kernels, hypervisors, device drivers, network protocols), each of which defines an interface that hides the implementation details of a particular set of functionality. Client programs built on top of each layer can be understood solely based on the interface, independent of the layer implementation. Despite their obvious importance, abstraction layers have mostly been treated as a system concept; they have almost never been formally specified or verified. This makes it difficult to establish strong correctness properties, and to scale program verification across multiple layers.

In this talk, I will present a novel language-based account of abstraction layers and show that they correspond to a strong form of abstraction over a particularly rich class of specifications which we call deep specifications. Just as data abstraction in typed functional languages leads to the important representation independence property, abstraction over deep specification is characterized by an important implementation independence property: any two implementations of the same deep specification must have contextually equivalent behaviors. We present a new layer calculus showing how to formally specify, program, verify, and compose abstraction layers. We show how to instantiate the layer calculus in realistic programming languages such as C and assembly, and how to adapt the CompCert verified compiler to compile certified C layers such that they can be linked with assembly layers. Using these new languages and tools, we have successfully developed multiple certified OS kernels in the Coq proof assistant, the most realistic of which consists of 37 abstraction layers, took less than one person year to develop, and can boot a version of Linux as a guest.

This talk represents joint work with Jeremie Koenig, Tahina Ramananandro, Zhong Shao, Newman Wu, Shu-Chun Weng, Haozhong Zhang, and Yu Guo. A paper documenting this result was published in POPL 2015. Both the paper, the technical report, and the presentation slides can be found at the URL: http://flint.cs.yale.edu/publications/dscal.html
Qualification of Formal Methods Tools
Darren Cofer (Rockwell Collins)

Abstract:
Formal methods tools have been shown to be effective at finding defects in and verifying the correctness of safety-critical systems such as avionics systems. The recent release of DO-178C and the accompanying Formal Methods supplement DO-333 will make it easier for developers of software for commercial aircraft to obtain certification credit for the use of formal methods.

However, there are still many issues that must be addressed before formal verification tools can be injected into the design process for safety-critical systems. For example, most developers of avionics systems are unfamiliar with which formal methods tools are most appropriate for different problem domains. Different levels of expertise are necessary to use these tools effectively and correctly. Evidence must be provided of a formal method’s soundness, a concept that is not well understood by most practicing engineers. Finally, DO-178C requires that a tool used to meet its objectives must be qualified in accordance with the tool qualification document DO-330.

The qualification of formal verification tools will likely pose unique challenges. To address these challenges, we have organized a Dagstuhl seminar (held the week before HCSS). The seminar will include sharing of knowledge from certification experts and formal methods researchers so that each can better understand the challenges and barriers to the use of formal methods tools.

Qualification is not a widely understood concept outside of those industries requiring certification for high-assurance. This presentation will provide background on tool qualification as it impacts the use of formal methods tools for aircraft certification, and include a “hot off the presses” report on results from the Dagstuhl seminar.
Abstract:

Formal methods have matured enough in the last decade to have made their way into several commercial tools for the development of safety-critical software. Recent revisions of software development standards by certifications authorities such as the FAA and EASA recognize this progress by providing explicit guidance on the use of formal methods in software development. For instance, the DO-333 Formal Methods Supplement of the DO-178C standard requires the production of evidence of logical soundness for any formal method used in the creation of the software system under certification. Any tools used to support the method must be themselves qualified according to a number of criteria.

This talk focused on our on-going investigations on tool qualification issues for modern SMT-based model checkers used to verify safety-properties of embedded software. Qualifying such tools is challenging because of their own complexity and that of the external SMT solvers they invoke. We will discuss a few approaches for developing certifying model checkers from certifying SMT solvers. Such tools produce proof certificates for each property they claim to have proved. In principle, these certificates can be checked by an external proof checker, with the effect of shifting the qualification effort to a much simpler tool. This solution is not realistic, however, because of the complexity of instrumenting a model checker to be fully certifying and because of scalability problems associated with fully detailed certificates. We will present a space of hybrid proof-engineering solutions where some components of the model checker are formally verified while others are proof-producing. We will report on our efforts with collaborators at Rockwell Collins and NYU to implement these solutions in the Kind 2 model checker and the CVC4 SMT-solver.

Certifying SMT solvers can also be incorporated into interactive theorem provers to increase these provers’ level of automation without compromising their high trustworthiness. We will also report on related work with collaborators at NYU and Inria to integrate CVC4 into Coq.
Multi-Language and Multi-Prover Verification with SAWScript

Aaron Tomb (Galois, Inc.)

Abstract:

SAWScript is a domain-specific language for orchestrating software analysis tasks, including formal proofs, that may involve multiple programming languages and multiple formal analysis tools, as part of the Software Analysis Workbench (SAW). SAWScript allows users to build and manipulate formal models representing the full semantics of code under analysis, from a variety of languages, including LLVM, JVM, and Cryptol. These formal models are represented within a single intermediate language, and can then be combined, transformed, and emitted to external proof tools such as SAT/SMT solvers and interactive theorem provers.

Using SAWScript allows an automated and integrated proof process, replacing what would otherwise have been a tedious and error-prone effort of manually combining results from a collection of subsidiary proof attempts. SAWScript is an integrated, interactive environment. It is not restricted to analyzing systems written in a single language, and one can compare, say C and Java implementations within the same SAWScript session. SAWScript also allows a variety of external solvers and proof tactics to be used within a single proof attempt. One could, for example, use one tool for reasoning about portions of the code involving arithmetic, and another for reasoning about bit-level semantics.

SAWScript has been used to verify equivalence of a number of production-quality algorithms, including implementations of Suite B cryptographic primitives from the OpenSSL and BouncyCastle libraries. This talk will demonstrate the use of SAWScript on a number of examples of verification and analysis.
CodeHawk: Sound Static Analysis for Proving the Absence of Memory Related Software Vulnerabilities

Douglas Smith (Kestrel Technology)

Abstract:
Most software vulnerabilities are due to coding errors. Testing is commonly the main means for detecting vulnerabilities, but testing alone only explores a small fraction of the possible behaviors of software. Sound static analysis is a technology that can examine source code and reason about all of its behaviors in order to detect coding errors that lead to vulnerabilities.

Kestrel Technology (KT) is developing a breakthrough static analysis tool, called CodeHawk, with an initial focus on memory related errors in C source code (covering 50 memory CWEs). CodeHawk is an advanced implementation of Abstract Interpretation, which works by automatically generating invariants for all code locations in a program and then using the invariants to prove safety and security conditions. CodeHawk also accepts human guidance in the form of strengthened invariants — the effect is to increase the percentage of memory accesses that are proved safe.

Currently CodeHawk exhibits near-perfect precision on standard benchmark codes produced by NIST and vulnerability experts. That is, when code contains a memory related error, CodeHawk will almost certainly report the error and will report few “false positives” (reports of code locations that are not errors). CodeHawk automatically provides mathematical proofs as evidence of safety and security claims for constructing an assurance case.

CodeHawk has been applied to a wide range of open-source C applications as part of a recent DHS-sponsored research project, and when run without application-specific preparation, CodeHawk can automatically prove 80%+ of memory accesses safe. For example, CodeHawk can currently prove 82% of memory accesses safe in the OpenSSL code that contains the HeartBleed vulnerability. KT researchers are working toward increasing the percentage of proved safe accesses to 100%. KT now believes it is practical to prove complete...
memory safety in a critical infrastructure program like OpenSSL by using CodeHawk with human guidance. Analysis time for proving automatically 82% of OpenSSL memory accesses safe was 9 hours using a single core PC but could be reduced to minutes using multiple cores.

The pressures of cyber-crime and cyber-espionage will lead to increased demands from customers for credible evidence of the absence of exploitable vulnerabilities and assurance that Heartbleed type attacks cannot be successful. CodeHawk is a front-runner in a new generation of automated tools that support cybersecurity analysis and the construction of assurance cases with mathematical proofs as evidence.
Detecting Malice in Commodity Software

Tim Fraser (DARPA)

Abstract:
The U.S. Government and commercial industry depend on commodity software. Unfortunately, the long supply chain that produces this software provides many opportunities for adversaries to insert backdoors or other hidden malicious functionality along the way. How can we gain confidence that our commodity software is free of malice? Conventional wisdom holds that this problem is so fundamentally difficult that no practical solution is attainable.

This talk will discuss three DARPA research programs aimed at overturning this conventional wisdom. Topics will include determining what malice to look for and where, ruling out the presence of malice, running reliable diagnostics on commodity devices that may be rigged to lie, the use of live competitive engagements between researchers and adversary teams to measure and drive technical progress, and the ability of coordinated research programs to gather research talent and direct it towards achieving a specific technical goal.
Remote Attestation for Cloud-Based Systems
Perry Alexander, Andy Gill, and Prasad Kulkarni
(University of Kansas)

Abstract:
Remote attestation provides a mechanism for gathering information from remote systems for assessing their trustworthiness. Using remote attestation an appraiser needing to assess trustworthiness makes a request to gather information from a target. The target’s attestation manager responds by executing an attestation protocol that gathers and bundles evidence that is sent to the appraiser. The appraiser un-bundles evidence and uses primary evidence to assess the target and meta-evidence to assess the target’s attestation process.

ArmoredSoftware enables remote attestation of processes in the cloud. Its components provide attestation, measurement, and appraisal capabilities that support remote attestation of and by applications running on virtual platforms managed by commodity cloud infrastructure. Each ArmoredSoftware application is bundled with its own appraisal, attestation, measurement, and vTPM components for performing trust establishment. Appraisers request information and negotiate attestation protocols with attestation agents. During negotiation, appraisers focus on needed information while attestation agents enforce their local privacy policies. Following negotiation, the attestation agent executes the agreed upon protocol that invokes measurement, vTPM, and nested appraisal services to gather and package evidence describing the application and its operational environment. The protocol and evidence are returned and together analyzed by the appraiser to determine trustworthiness.

Remote attestation in a cloud-managed virtual environment introduces a number of specific challenges that ArmoredSoftware must address. These include, but are not limited to: opacity and variability of execution environment details; maintaining links to hardware roots-of-trust across migration; measuring the diversity of user-space applications; and bundling evidence from multiple sources.
Our talk will focus on the ArmoredSoftware architecture, protocol semantics, and experimental implementation. We will first outline the ArmoredSoftware architecture focusing on interactions among major components. We will then discuss specifics of the first-class, monadic protocol representation at the center of attestation and measurement focusing on a commonly used certificate authority-based protocol example. Finally, we will discuss results from experiments on early prototype implementations. An informal demonstration will be available before and after the presentation.

*This work was sponsored in part by the United States Department of Defense*
Software Defenses
Inspired by Biodiversity
Michael Franz (University of California, Irvine)

Abstract:

Today’s software monoculture creates asymmetric threats. An attacker needs to find only one way in, while defenders need to guard a lot of ground. Adversaries can fully debug and perfect their attacks on their own computers, exactly replicating the environment that they will later be targeting.

One possible defense is software diversity, which raises the bar to attackers. A diversification engine automatically generates a large number of different versions of the same program, potentially one unique version for every computer. These all behave in exactly the same way from the perspective of the end-user, but they implement their functionality in subtly different ways. As a result, a specific attack will succeed on only a small fraction of targets and a large number of different attack vectors would be needed to take over a significant percentage of them. Because an attacker has no way of knowing a priori which specific attack will succeed on which specific target, this method also very significantly increases the cost of attacks directed at specific targets.

We have built such a diversification engine, which is now available as a prototype. We can diversify large software distributions such as the Firefox and Chromium web browsers or a complete Linux distribution. We will present our overall system architecture and preliminary insights and measurements. We will also discuss some practical issues, such as the problem of reporting errors when every binary is unique.

Finally, we will report on a set of groundbreaking new software diversity techniques that can additionally also defend against side-channel attacks by dynamically and systematically randomizing the control flow of programs. Previous software diversity techniques transform each program trace identically. Our new technique instead transforms programs to make each program trace unique. This approach offers probabilistic protection against both online and off-line side-channel attacks, including timing and cache-based attacks.
In particular, we create a large number of unique program execution paths by automatically generating diversified replicas for parts of an input program. At runtime we then randomly and frequently switch between these replicas. As a consequence, no two executions of the same program are ever alike, even when the same inputs are used. Our method requires no manual effort or hardware changes, has a reasonable performance impact, and reduces side-channel information leakage significantly when applied to known attacks on AES.
David Kaloper Meršinjak and Hellmut Mehnert
(University of Cambridge)

Abstract:
Transport Layer Security (TLS) implementations have a history of security flaws. The immediate causes of these range from simple programming errors, such as memory management, to subtle violations of the protocol logic. Deeper causes can be seen in the challenges of interpreting the ambiguous prose specification, the complexities inherent in large APIs and code bases, unsafe performance-oriented programming choices, and the impossibility of directly testing conformance between implementations and the specification.

We present nqsb-TLS, the result of our re-engineering approach to improve the quality of security protocol implementations. The same code serves dual roles: it is both a specification of TLS, executable as a test oracle to check conformance of traces from arbitrary implementations, and a secure and usable executable implementation of TLS. nqsb-TLS employs a modular and declarative programming style that also allows it to be compiled into a Xen unikernel (a specialised virtual machine image) with a TCB that is 2.5% of a standalone system running a standard Linux/OpenSSL stack, with all network traffic being handled in a memory-safe language.

nqsb-TLS focuses on protocol-level interoperability, and makes no effort to be compatible with existing (and often poorly designed) library APIs such as OpenSSL. The higher-level API in nqsb-TLS makes it harder to misuse the library, and is demonstrated via several unikernel applications ranging over HTTPS, IMAP, Git, Websocket clients and servers.

This is joint work by David Kaloper Mersinjak, Anil Madhavapeddy, Hannes Mehnert and Peter Sewell. It is funded in part by the EPSRC REMS grant (Rigorous Engineering of Mainstream Systems), EP/K008528/1.
Rigorous Architectural Modelling for Production Multiprocessors

Shaked Flur, Kathryn Gray, *Gabriel Kerneis, **Luc Maranget, Christopher Pulte, ***Susmit Sarkar, and Peter Sewell (University of Cambridge, * Google, ** Inria, *** University of St Andrews)

Abstract:

Processor architectures are critical interfaces in computing, but they are typically defined only by prose and pseudocode documentation. This is especially problematic for the subtle concurrency behaviour of weakly consistent multiprocessors such as ARM and IBM POWER: the traditional documentation does not define precisely what programmer-observable behaviour is (and is not) allowed for concurrent code; it is not executable as a test oracle for pre-Silicon or post-Silicon hardware testing; it is not executable as an emulator for software testing; and it is not mathematically rigorous enough to serve as a foundation for software verification.

We present a rigorous architectural envelope model for IBM POWER multiprocessors, and similar work in progress for ARM, that aims to support all of these for small-but-intricate test cases, integrating an operational concurrency model with an ISA model for the sequential behaviour of a substantial fragment of the user-mode instruction set (largely automatically derived from the vendor pseudocode, and expressed in a new ISA description language). The key question is the interface between these two: we have to allow all the required relaxed-memory behaviour, without overcommitting to some particular microarchitectural model. Our models can be automatically translated into executable code, which, combined with front-ends for concurrency litmus tests and ELF executables, can interactively or exhaustively explore all the allowed behaviours of small test cases.

This is joint work by S. Flur, K. Gray, G. Kerneis*, L. Maranget**, C. Pulte, S. Sarkar***, and P. Sewell, University of Cambridge (* Google, ** Inria, *** St Andrews). It is funded in part by the EPSRC REMS grant (Rigorous Engineering of Mainstream Systems), EP/K008528/1.
A Formal Specification of x86 Memory Management
Shilpi Goel and Warren Hunt (UT Austin)

Abstract:
We are developing a formal and executable x86 ISA specification that includes memory management via paging and segmentation. We regularly use this specification to mechanically verify user-level x86 machine-code programs. Our recent efforts have been devoted to developing a theory for reasoning about system-level programs that are aware of memory management mechanisms like paging.

On contemporary x86 implementations, every linear (virtual) address is mapped to a physical address by hierarchical, page-table data structures. We have been analyzing the effect of address mapping on the state of the memory management system. As a part of this effort, we proved that multiple translations of a specific linear address produce the same physical (translated) address. This property is surprisingly complicated because performing a translation may modify the page-table data structures. Establishing properties about the translation mechanism is essential if we are to verify programs such as the memory management code of an operating system.
Language-Based Hardware Verification with ReWire: Just Say No! to Semantic Archaeology

William Harrison and Adam Proctor
(University of Missouri)

Abstract:

There is no such thing as high assurance without high assurance hardware. High assurance hardware is essential, because any and all high assurance systems ultimately depend on hardware that conforms to, and does not undermine, critical system properties and invariants. And yet, high assurance hardware development is stymied by the conceptual gap between formal methods and hardware description languages used by engineers.

This talk presents ReWire, a Haskell-like functional programming language providing a suitable foundation for formal verification of hardware designs, and a compiler for that language that translates high-level, semantics-driven designs directly into working hardware. ReWire’s design and implementation are presented, along with a case study in the design of a secure multicore processor based on the Xilinx PicoBlaze, demonstrating both ReWire’s expressiveness as a programming language and its power as a framework for formal, high-level reasoning about hardware systems.

The philosophy that drives the development of ReWire may be summed up as the conviction that semantic archaeology is the bane of high assurance computing. By “semantic archaeology”, we mean the process of developing a formal specification for an existing computing artifact. Semantic archaeology is time-consuming and expensive, because such artifacts are rarely written with formal semantics in mind, and, consequently, the formal methods scientist must attempt a painstaking reconstruction of the system semantics from informal and often incomplete natural language documents (if, indeed, such a reconstruction is even possible).

In keeping with the proof engineering theme of HCSS 2015, this work aims to
engineer maintainable, extensible and reusable proofs via a language design approach. Semantic archaeology is avoided by starting from a semantically-defined language. ReWire is both a computational $\lambda$-calculus suitable for writing formal specifications and an expressive functional language and compiler for generating efficient hardware artifacts. With ReWire, the text of a design is verified (rather than a reconstructed model of the design) and the compiler transforms that same design into hardware, thereby unifying the languages of specification, design and implementation. The hypothesis of this work is that this duality will position ReWire to avoid the pitfalls of semantic archaeology without sacrificing performance. To date, our case studies have generated artifacts with acceptable performance.

The research presented here is joint work with Professor Michela Becchi and Ian Graves of the University of Missouri and Dr. Gerard Allwein of the US Naval Research Laboratory.

References


CONFERENCE PRESENTATIONS

Achieving High Speed and High Assurance in a Hardware-Based Cross-Domain System Using Guardol
David Hardin and Konrad Slind (Rockwell Collins)

Abstract:
As military systems become increasingly internetworked, they become ever more vulnerable to cyber attacks. Experience with the public Internet has shown that high-profile vulnerabilities, such as Shellshock, POODLE, HeartBleed, MyDoom, Sobig.F, Conficker, and the like, continue to outpace the ability of the off-the-shelf software development community to “plug the leaks”. These vulnerabilities are pernicious in that they are generally very low-level, and are “a needle in a haystack,” existing within a software corpus of millions of lines of code. In context of the public Internet, patches can be developed rather quickly (on the order of days to weeks), and distributed throughout the Internet in a matter of days. Networked military systems typically utilize the same sorts of operating systems and applications as in the public Internet, but do not embrace rapid update cycles; a shipboard system, for example, may go years between upgrades, leaving many known vulnerabilities unpatched for long stretches of time.

Cross-domain systems (or guards) can provide a first line of defense against cyber attack. In a typical guard, both the “internal” and “external” networks terminate at the guard, and only certain protocols are allowed to transit the guard from one side to the other. Further, user-programmable logic allows the guard to accept, reject, or modify data payloads carried by the allowed protocols, based on some user-defined criteria. In theory, cross-domain systems provide a very high “cyber-wall” to any would-be attacker. However, in practice most cross-domain systems are architected as applications hosted on the same operating systems and hardware platforms that have shown to be highly vulnerable to cyber attack. Thus, it is quite possible for an attacker to exploit a known vulnerability in the cross-domain system platform in order to bypass the guard logic entirely.

In order to address these customer concerns, we are developing a high-speed, high-assurance hardware-based guard. Implementing the guard core
using FPGA technology allows us to maintain programmability of the guard, providing a much higher resistance against cyber attack, while also producing very high speed operation. We isolate the user from the details of hardware programming by producing a VHDL code generator for the Guardol programming language. Guardol provides the user with a high-assurance programming environment, in which user-written property specifications can be automatically proven using the Guardol toolchain.

We have focused our initial efforts on regular-expression based hardware guards, for which the Guardol toolchain generates a mathematically verified, high-speed DFA implementation. We will demonstrate the feasibility of our approach on FPGA-based guard hardware.
Bringing Hardware Hacking to Life
Colin O’Flynn (Dalhousie University)

Abstract:
But is it practical? This question is always banded about to dismiss some research publication showing an advanced attack such as side-channel power analysis or glitching. Most of these publications are using university-backed labs or expensive commercial equipment, leading to the assumption an attacker would also require such equipment.

But this assumption is false – the attacks can be applied with a few hundred dollars worth of equipment. And they can be applied to real devices, effectively breaking otherwise secure encryption algorithms. It’s not enough to have correctly implemented AES-256 in a bootloader for example; if the AES implementation is a standard off-the-shelf implementation, it might be possible to recover the encryption key from a bootloader with minimal work.

This presentation details both the theoretical side and practical side of side-channel power analysis attacks, with a quick detour into fault injection. The presenter has worked extensively on the open-source ChipWhisperer project, which aims to teach embedded and software engineers about the vulnerabilities hiding in what are assumed to be secure algorithms.

As we design secure systems for the future, resistance against fault injection and side-channel power analysis will become more important than ever. While such attacks are not yet a regular occurrence, they will undoubtedly become the next lowest hanging fruit to be picked once a better job of patching simple software vulnerabilities is performed. But many of these vulnerabilities cannot be patched with a firmware update – they require costly changes to hardware.

This presentation details not only the technical workings of side-channel power analysis and glitching attacks, but also how they apply to real systems, and what this means to those designing those systems. All the tools used in this presentation are open-source, giving attendees the ability to dive into more details and try their hand at power analysis and glitching attacks.
Building Privacy-Aware Computing Systems: An Overview of Current Capabilities and Technical Challenges

Shantanu Rane (PARC)

Abstract:
Big data analytics has the potential to solve many of the world’s pressing problems and to create exciting new opportunities for individuals, corporations and governments. Its applications include finding treatments and cures for diseases, streamlining the world’s transportation systems, securing people and infrastructure against acts of terrorism, developing new mechanisms for energy distribution and pricing in smart grids, driving customer-centric businesses in the internet age, and many more. However, the big data requirement appears, almost fundamentally, to be in conflict with the idea of privacy. Indeed, much of big data analytics today involves indiscriminate information gathering with scant regard for individual privacy. How can expressive data analysis be conducted while protecting the privacy of people on whom that data is collected?

In this talk, I will present an overview of the area of privacy-aware computing. The objective is to foster discussions on privacy-aware computing capabilities and the tradeoffs we have to make while using them. I will briefly describe the main players in the big data analytics setting (data owners, data controllers, data users) and discuss their incentives and privacy requirements. Next, I shall describe technical solutions for privacy-aware computing, including beautiful results from cryptography (homomorphic encryption, secure multiparty computation, verifiable computing), and statistical privacy mechanisms (k-anonymity and its variants, differential privacy). For each technique, I shall point out the gap between its capabilities and the requirements imposed by practical systems. These gaps expose several interesting challenges that motivate new research in privacy technologies.
Abstract:

We need cryptographic software that we can trust. However, our trust in widely used cryptographic software has been repeatedly undermined by implementation bugs and side-channel attacks. At the heart of these issues lies an uncomfortable gap between provable security and practical cryptography; in short, provable security has been immensely successful in analyzing the security of cryptographic constructions, but relies on an idealized model (the computational model of cryptography) that elides many common causes of attacks. Real-world cryptography aims to address this problem, by applying the principles of provable security to analyze widely used protocols under the light of more precise security definitions and adversary models. This is an essential step towards making provable security practically relevant, but models from real-world cryptography remain too abstract to capture side-channel attacks and implementation bugs. In the talk, I will outline a further step to narrow the gap between provable security and practical cryptography. Our approach builds on our earlier work in computer-aided cryptography, and in particular on EasyCrypt, a computer-aided tool which supports deductive verification of reductionist proofs in cryptography and has been used to verify prominent examples of cryptographic constructions and protocols, including PKCS-OAEP encryption and the Naxos protocol, and is currently applied to multi-party and verifiable computation, and to voting systems. In addition, our approach uses some recent development in programming languages and formal verification, and more specifically certified static analyses, verified compilers, and type systems and program logics for low-level languages.

In its simplest form, our approach consists of three steps: i. verify black-box security of a cryptographic system in the computational model, using tools for computer-aided security like EasyCrypt; ii. carry black-box security to assembly implementations, using verified compilers; iii. prove that the assembly implementation does not leak, using certified static analysis. I will illustrate the effectiveness of our approach by instantiating it to the notable cases of...
CONFERENCE PRESENTATIONS

masked implementations (protection against DPA attacks) and constant-time implementations (protection against cache attacks). Focusing on preventing leakage of implementations, I will present the principles and experimental results for a new verifying compiler that generates efficient masked implementations of blockciphers (AES, Simon... ) at arbitrary orders, and uses a fine-grained information flow type system to verify that the generated code is secure. The correctness of the certifying compiler rests on a deep connections between leakage in the threshold probing model of Ishai, Sahai and Wagner (which has recently been proved equivalent to the nosiy leakage model of Chari et al), and a generalization of probabilistic non-interference, and on a stronger notion of non-interference, that justifies the compositional reasoning of the type system.

I will further provide a broader perspective on applications of our approach to more complex cryptographic systems, and with challenges for verified programming languages and proof engineering.

For further background information, please consult: www.easycrypt.info.
A Cut Principle for Information Flows

Joshua Guttman and Paul D. Rowe
(The MITRE Corporation and
Worcester Polytechnic Institute)

Abstract:
We view a distributed system as a graph of active locations with unidirectional channels between them, through which they pass messages. In this context, the graph structure of a system constrains the propagation of information through it.

Suppose a set of channels is a cut set between an information source and a potential sink. We prove that, if there is no disclosure from the source to the cut set, then there can be no disclosure to the sink. We introduce a new formalization of partial disclosure, called blur operators, and show that the same cut property is preserved for disclosure to within a blur operator. This cut-blur property also implies a compositional principle, which ensures limited disclosure for a class of systems that differ only beyond the cut.
Models and Games for Quantifying Vulnerability of Secret Information

Piotr Mardziel (University of Maryland)

Abstract:
Quantitative information flow (QIF) is concerned with measuring the amount of secret information that leaks through a system’s observable behavior during its execution. The system takes secret (high) input and produces (low) output that can be observed by an adversary. Before the system is run, the adversary is assumed to have some a priori information about the secret. As the system executes, the adversary’s observations are combined with knowledge about how the system works, resulting in some a posteriori information about the secret. A general principle of QIF states that the leakage of information by the execution is defined as the increase in the adversary’s information.

Past work has studied how to precisely instantiate this principle, considering various notions of information and how they relate to each other, and increasingly powerful adversaries. For example, active adversaries may be allowed to provide (low) inputs to the system, to manipulate it to leak more data, and adaptive adversaries may choose these inputs based on the observable behavior of the system. Most approaches to QIF are limited in three regards: 1) assumption of static (unchanging) secrets, 2) focus only on the goals of the adversary (as opposed to the defender or secret holder), and 3) consideration of only passive defenders. This talk will summarize our recent works that begin to address these short-comings.
Abstract:
The Director of Civil Liberties and Privacy is the primary advisor on protection of civil liberties and privacy to NSA’s Director and is the lead for promoting and integrating civil liberties and privacy protections into NSA policies, plans, procedures, technology, programs, and activities. The Office of Civil Liberties and Privacy is responsible for developing meaningful civil liberties and privacy processes and is working to identify and study methodologies that take a data-driven and use-driven approach to evaluating privacy impact.
High Assurance Cryptography Synthesis with 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. As summarized last year at HCSS, Galois has “rebooted” Cryptol and created, from the ground up, a new Open Source Cryptol release: Cryptol version 2.

A new major feature under development within Cryptol 2 and the Galois Software Assurance Workbench (SAW) is a set of new backends capable of synthesizing formally verified implementations and test benches for arbitrary cryptographic algorithms. Within Cryptol 1 we were previously able to synthesize C, JVM, and VHDL implementations, but without any additional artifacts to provide evidence of the implementations’ correctness. Cryptol’s new backends will target (at least) C (directly and via LLVM), JVM, VHDL, and (System) Verilog, along with Cryptol-independent evidence of the synthesized code’s correctness.

In particular, we are developing a new subsystem called the “Structured VC Generator” that produces evidence in a form amenable to analysis by other tools. More specifically, it accepts as input a set of Cryptol properties and a Cryptol specification. Using the SAW symbolic interpreter it generates, for each Cryptol function reachable from the properties, a complete set of verification conditions characterized as input-output relations on said function. This is a particularly powerful technique for guaranteeing functional correctness because top-level theorems, which are expressed as universally quantified sentences, derive new theorems at function application boundaries. This results in a enormously rich set of input-output relations useful for runtime verification.

The generated VCs, which are encoded in the IR of SAW, can then be pushed through the backends to automatically generate test benches, amenable to
runtime or static verification, for both software and hardware implementations. Theorems about implementations are stated in the natural fashion given the backend in question; C specifications are captured in ACSL, JVM specifications in JML, and VHDL/(System) Verilog specifications in HDL assertions. Consequently, all generated implementations have full formal specifications and complete test benches for validation and verification, and their correctness and security properties can be independently analyzed using other tools capable of reasoning about C, LLVM, JVM, VHDL, and System Verilog.
Privacy through Accountability
Anupam Datta (Carnegie Mellon University)

Abstract:
Privacy through accountability refers to the principle that entities that hold personal information about individuals are accountable for adopting measures that protect the privacy of the data subjects. In this talk, I will cover computational treatments of this principle. This emerging research area, which my research group has played a pivotal role in developing, has produced precise definitions of privacy properties and computational accountability mechanisms to aid in their enforcement. After providing an overview of the research area, I will focus on two of our recent results in Web privacy.

First, I will present our joint work with Microsoft Research on building and operating a system to automate privacy policy compliance checking in Bing. Central to the design of the system are (a) LEGALEASE — a language that allows specification of privacy policies that impose restrictions on how user data is handled; and (b) GROK — a data inventory for Map-Reduce-like big data systems that tracks how user data flows among programs. GROK maps code-level schema elements to datatypes in LEGALEASE, in essence, annotating existing programs with information flow types with minimal human input. Compliance checking is thus reduced to information flow analysis of big data systems. The system, bootstrapped by a small team, checks compliance daily of millions of lines of ever-changing source code in the data analytics pipeline for Bing written by several thousand developers.

Second, I will describe the problem of detecting personal data usage by websites when the analyst does not have access to the code of the system nor full control over the inputs or observability of all outputs of the system. A concrete example of this setting is one in which a privacy advocacy group, a government regulator, or a Web user may be interested in checking whether a particular web site uses certain types of personal information for advertising. I will present a methodology for information flow experiments based on experimental science and statistical analysis that addresses this problem, our tool AdFisher that incorporates this methodology, and findings of opacity, choice and discrimination from our experiments with Google’s Ad EcoSystem.
Private Disclosure of Information
Daniel Aranki and Ruzena Bajcsy (UC Berkeley)

Abstract:
The healthcare domain is prime for investigating privacy. Any information about the subject’s physical, cognitive and/or mental states is private. A medical technology with increasing attention is health tele-monitoring, where a technology is used to collect health-related data about patients, which are later submitted to a medical staff for monitoring. The data are then used to assess the health-status of patients and provide them with feedback and/or intervention.

We are inundated with different body sensors that can monitor physical and physiological states of the subject. These sensors can report to the subject who wears them but they can also wirelessly report the data to the healthcare station in a tele-monitoring fashion. This configuration makes health tele-monitoring a privacy-sensitive cyber-physical system. In the phase of communication, there is a serious privacy risk since an adversary who can intercept the communication, can also misuse the data. In order to address the privacy risk in the aforementioned scenario, we devised a technical framework that aims to protect the privacy of individuals during communication, called Private Disclosure of Information (PDI).

PDI is aimed to prevent an adversary from inferring certain sensitive information about subjects using the data that they disclosed during communication with an intended recipient. PDI is an information-theoretic, statistical framework, which introduces a view of considering the interpretation of data, through inference, in maintaining privacy. PDI considers the case where each subject belongs to a private class (such as a health-condition or a disease). The subject-class membership can be statistically inferred from the data that the subject discloses during communication (data such as vital signs, symptoms, sensory data, etc...). PDI considers cases, such as tele-monitoring, where this class membership is known to both the subject herself and to the intended recipient, but not to the adversary. PDI leverages this advantage in knowledge in order to minimize the amount of information leaked by the communication to an adversary, who is trying to infer the class membership of subjects from the communicated data.
We leverage encoding schemes, where the encoding function of messages is different per class, for the communication. Out of all such schemes, PDI uses the encoding scheme that minimizes the amount of information which the transmitted messages carry to inferring the private class. PDI adopts the Mutual Information measure to model this quantity. The framework results in a learning problem for the encoding scheme, which we implemented in a form of a MATLAB toolbox.

In this talk, we present the technical details of the PDI framework, in addition to technical results on its privacy guarantees. We present the resulting learning problem of the framework. We also present a sufficient condition that guarantees perfect privacy, regardless of the adversary’s auxiliary knowledge, while preserving full utility of the information to the intended recipient. We demonstrate this result with specific examples and demonstrate the applicability of PDI on a real-world data set that simulates a health tele-monitoring scenario.
The poster session will be held between 10:30 a.m. - 11:15 a.m. and 3:00 p.m. – 3:45 p.m. on Thursday, May 7, in the atrium of the Governor Calvert House.

SecProve: Analyzing Software for Security During Construction
Myla Archer, U.S. Naval Research Laboratory

Towards Formalizing Software Bugs
Paul E. Black, NIST

FUSE: Beyond Single App Security
Rogan Creswick, Galois, Inc.

Science of Security Publications of Interest
Don Goff, Cyberpack Ventures

Deep Specifications and Certified Abstraction Layers
Ronghui Gu, Yale University

A Meta-Model for the Assessment of Systems
Jennifer Guild, Navy

The End-to-End Verifiable Internet Voting Project
Joseph Kiniry, Galois, Inc.

CoqPIE: A Coq IDE Aimed at Improving Proof Development Productivity
Kenneth Roe, The Johns Hopkins University

Runtime Assurance for Complex Autonomy
John Schierman, Barron Associates, Inc.

Inferring Contracts and Proving Properties Using Abstract Interpretation Over a Global Value Numbering
Tucker Taft, AdaCore

Bringing Roots of Trust to Reality
Adam Wick, Galois, Inc.
SecProve: Analyzing Software for Security During Construction

Myla Archer, Elizabeth I. Leonard, and Constance L. Heitmeyer
(U.S. Naval Research Laboratory)

Abstract:

SecProve is a prototype tool, designed and currently being implemented at the Naval Research Laboratory, for analyzing software for global security properties using local, automatically generated code assertions. This technique permits SecProve to handle properties beyond the application-independent ones (e.g., “no buffer overflow”) treated by other tools. SecProve’s generated assertions are of two classes: Class 1 (known to be valid or derivatively valid) and Class 2 (desired to be valid). Verification of a global property is typically done by verifying the Class 2 assertions generated from that property. A Class 2 assertion can be verified using Class 1 assertions plus verification of any Class 2 assertions on which those Class 1 assertions depend. SecProve automatically generates assertions based on three sources: 1) developer annotations and function/procedure contracts (provided by the developer or automatically generated themselves), 2) the source code itself, using forward propagation, and 3) property specifications. The use of forward propagation of assertions allows code to be analyzed as, rather than after, it is constructed.

To make it possible to generate assertions from a property, SecProve provides templates for specifying properties of certain classes. Each template is accompanied by its own assertion-generation scheme. Thus, addition of a new property template to SecProve requires the extension of SecProve’s overall assertion generation scheme. For each template, assertions are generated in two phases. The first phase, which generates mostly those Class 2 assertions on which the property depends, is done by placement of assertions. The second phase, which mostly generates Class 1 assertions, uses forward propagation of assertions.

Any property specification template included in SecProve may include slots for one or more predicates or functions that will be used to capture history information. Currently, SecProve has templates for properties from three classes: dataflow, sanitization, and nonbypassability. The template for the
sanitization property has a slot for a sanitized predicate that, applied to variable \( x \), is used to express that \( x \) is “clean” as opposed to “dirty”. Used by every dataflow property is a to function built into SecProve that, applied to a program variable \( x \), captures the set of program variables from which content has flowed to \( x \). The semantics of the predicates and functions is defined not explicitly, but implicitly through the placement and forward propagation schemes for assertions involving the values of these predicates and functions. Thus, forward propagation updates to and sanitized in the “obvious” way to reflect their intended meanings.

In programs with sufficiently complex conditional branches, it may be necessary to incorporate a mechanical theorem prover into SecProve to perform assertion checking. However, in our examples so far, many of the Class 2 assertions to be checked capture simple history, and heavy duty theorem proving is not needed to check them. Our current focus is on implementing straightforward assertion checking for simple history assertions.

The architecture of SecProve—which includes a source code preprocessor, the assertion generator, the assertion checker, a GUI, and a database through which these components communicate—has been described in [1]. A patent application [2] was filed for SecProve in January, 2014. This poster will focus on the current status of (and plans for) SecProve and walk through a user scenario.


Abstract:
High-confidence systems must not be vulnerable to attacks that impact the security, reliability, or availability of the system as a whole. That is, they must have no software weaknesses (bugs) that could result in a vulnerability. A vulnerability is “a weakness in system security requirements, design, implementation, or operation that could be accidentally triggered or intentionally exploited and result in a security failure.” [1] The Common Weakness Enumeration (CWE) [2] is a collection of known software weakness types described in plain English with demonstrative examples, relations to other CWEs, and some formal definitions. It is a considerable community effort, but many of the descriptions are inaccurate, incomplete, inconsistent, or ambiguous. We discuss our vision of precisely defining software bugs: wholesale restructuring of CWEs, incorporating Software Fault Patterns (SFPs) and Semantic Templates, and formalizing them. This will enable automatic generation of software testing components and tools, software analysis tools that always find these weaknesses, formal proofs of systems properties, assuring absence of particular vulnerabilities, and mitigating vulnerabilities by filtering out exploiting attacks. Accurate and precise definitions of software weaknesses, along with their causes and consequences, are essential for constructing formal proofs of presence or absence of such weaknesses in a program.

We picked three CWEs that we thought would be representative and would help expose the inherent opportunities and challenges: CWE-307 Improper Restriction of Excessive Authentication Attempts; CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer, and CWE-78 Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’). The three selected CWEs are quite different from each other in aspects from source code footprints to ways of exploitation.

We propose new definitions for these three. Our definitions are substantially more precise and accurate than the current CWE descriptions. For instance, the current description summary of CWE-307 includes failure to prevent
“multiple failed authentication attempts within a short time frame”. However, that description does not provide precise meanings for “multiple” and “short”. Our new definition recognizes that CWE-307 actually represents a set of weaknesses, each of which satisfies particular institution-specific definitions of “multiple” and “short”. Next, the description summary of CWE-119 is “The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside the intended boundary of the buffer”. However, “read from or write to a memory location” is not tied to the buffer. Our definition clarifies that access is through the same buffer to which the intended boundary pertains. Our definition also accurately, precisely, and concisely describes violation of memory safety. Finally, the description summary of CWE-78 is based on the software “using externally influenced input”, and incorrectly neutralizing “special elements that could modify the intended OS command”. “Using input”, “intended command”, and “correctly neutralizing” are imprecise. We precisely define “using input” and “intended command”. We do not include “correctly neutralizing”, because it simply means that intended OS command cannot be modified.

We also explain some of our work toward formalizing definitions [3] and restructuring and unifying CWEs, SFPs, and Semantic Templates, including causes and consequences.


CONFERENCE POSTERS

FUSE: Beyond Single App Security
Rogan Creswick (Galois, Inc.)

Abstract:
FUSE is a tool to help security analysts see how a collection of Android applications will operate together on a device. FUSE combines information from static analysis of all the individual apps provisioned to a mobile device to give security analysts the ability to quickly see where collusion or other app interaction may occur—interactions that no single-app analysis can discover. Analysts can combine the information from FUSE’s visual data flow graphs with their expert domain knowledge to focus their attention on the applications that pose the greatest risk. FUSE allows analysts to interactively filter different types of data flow, select subsets of apps, and even reveal the specific instructions where inter-app data transfer occurs. Because FUSE analyzes compiled Android APKs, the analysis does not require access to application source code.

Managing information to ensure the privacy of your personal, corporate, or organizational information requires more than a single-app view of security. The big-picture analysis possible with FUSE provides a high-level view of how personal and confidential information is handled on an Android device. Our approach to Android analysis starts with that high-level view of application collections, but also allows for deep dives into decompiled application bytecode through an integrated ‘dex explorer’.

This poster builds on previous talks at HCSS in 2012 and 2014, showing the latest state of the FUSE tools for single- and multi-app Android analysis. It showcases changes from the previous presentations, including: support for automated policy evaluation, interactive deep-dives, and the application of information flow slicing techniques to identify undesirable flows.
Science of Security
Publications of Interest
Don Goff (Cyberpack Ventures)

Abstract:
The Science of Security project supports research efforts currently underway at the nations universities and determines how that research relates to the development and maturation of the science of cybersecurity.

The project identifies promising research fields and how they relate to the creation of a broad scientific body of knowledge and the development of basic precepts in this new field. The poster shows the projects content from the CPS-VO Science of Security web page, including opportunities available to researchers, current bibliography, and items of interest to the community.
Deep Specifications and Certified Abstraction Layers
Ronghui Gu (Yale University)

Abstract:
Modern computer systems consist of a multitude of abstraction layers (e.g., OS kernels, hypervisors, device drivers, network protocols), each of which defines an interface that hides the implementation details of a particular set of functionality. Client programs built on top of each layer can be understood solely based on the interface, independent of the layer implementation. Despite their obvious importance, abstraction layers have mostly been treated as a system concept; they have almost never been formally specified or verified. This makes it difficult to establish strong correctness properties, and to scale program verification across multiple layers.

In this talk, I will present a novel language-based account of abstraction layers and show that they correspond to a strong form of abstraction over a particularly rich class of specifications which we call deep specifications. Just as data abstraction in typed functional languages leads to the important representation independence property, abstraction over deep specification is characterized by an important implementation independence property: any two implementations of the same deep specification must have contextually equivalent behaviors. We present a new layer calculus showing how to formally specify, program, verify, and compose abstraction layers. We show how to instantiate the layer calculus in realistic programming languages such as C and assembly, and how to adapt the CompCert verified compiler to compile certified C layers such that they can be linked with assembly layers. Using these new languages and tools, we have successfully developed multiple certified OS kernels in the Coq proof assistant, the most realistic of which consists of 37 abstraction layers, took less than one person year to develop, and can boot a version of Linux as a guest.

This talk represents joint work with Jeremie Koenig, Tahina Ramananandro, Zhong Shao, Newman Wu, Shu-Chun Weng, Haozhong Zhang, and Yu Guo. A paper documenting this result was published in POPL 2015. Both the paper, the technical report, and the presentation slides can be found at the URL:
http://flint.cs.yale.edu/publications/dscal.html
A Meta-Model for the Assessment of Systems

Jennifer Guild (Navy)

Abstract:

In this poster, we will provide an overview of a methodology that maps mathematical models to assessment evidence, including formal proofs, in a human friendly manner for certification processes. This is relevant to HCSS as it provides an assessment methodology that integrates formal proofs without requiring the assessor to be an expert in formal modeling. The methodology (meta-model) provides a way for the assessor to represent, organize, and refine the various aspects of the assessment: the flaws, countermeasures, threats, vulnerabilities, risk, etc. This provides a level of assessment detail not provided in the current approaches, a more precise basis for the properties that must be maintained in the operational system and the ability to reuse the assessment of a system by another entity without reassessing the system.

National Security Systems are assessed to determine our confidence in their level of robustness, where robustness is the characterization of strength of a security function, mechanism, service, or solution, and the assurance that it is implemented and functioning correctly. For National Security Systems, current practice requires a single assessment that does not address formal proofs to be conducted against a system’s implementation. The current approach combines two types of assessments. The first, the technical assessment, is conducted on an implementation in a laboratory environment and is conducted prior to placement of the system in the operational setting. The second, the operational site assessment, is conducted once the system is implemented at the operational site, and may include physical connection to live networks.

Currently, there is no assessment methodology in use by the US government that provides a model and methodology for assessing technical and operational evidence and risk individually. Such a methodology and models would provide a level of reciprocity not available reducing assessment costs and allow better reuse of systems.

We propose a methodology that revisits individual models to enable the assessor to represent their knowledge of the system’s capabilities and correlate the models to the evidence, including formal proofs. The content of these models is refined from generalized to specific as the assessment progresses.
The individual models will, in combination, comprise the meta-model, mapping the completed models to the evidence of the assessment.

We propose the use of mathematical models for a number of aspects that an assessor must consider when assessing a system, regardless of its complexity or connectivity. Mathematical models detail the flaws, countermeasures, vulnerabilities, threats, probabilities, attack vectors, impacts, and risk on the operational environment. Vulnerabilities are defined as those flaws with perceived, partial, or no countermeasures. The impact is in terms of the value of the asset with the vulnerabilities and the associated threats.
The End-to-End Verifiable Internet Voting Project
Joseph Kiniry (Galois, Inc.)

Abstract:
Voting systems have strict security and privacy requirements, which are different from those in many other domains. They must not expose information that connects voters and their votes; moreover, to prevent voter coercion, individual voters must not themselves be able to expose information that proves how they voted. These requirements are fundamentally in tension with exposing enough information to determine that the counted votes match the cast votes, and are difficult to fulfill even in purely physical voting systems.

Over the past two decades, various forms of Internet-based voting have been proposed and carried out in the United States and other countries. A primary motivation for Internet voting is accessibility, both for voters with disabilities and for voters who cannot easily visit polling places or submit absentee ballots in a timely fashion (e.g., expatriates, those serving abroad in the military, etc.).

Unfortunately, though well-intentioned, these Internet voting systems have generally been fraught with security and privacy issues. Some systems used in national-level elections have been subsequently found to leak voter and vote information and to be susceptible to various forms of manipulation that can untraceably alter election results. This has led many activists and technologists to write off Internet voting as inherently insecure and unsuitable for use in real elections.

In December 2013 the Overseas Vote Foundation (OVF) announced the End-to-End Verifiable Internet Voting Project (E2E VIV). Its goal is to study the feasibility of developing an Internet-based system that fulfills the security and privacy requirements associated with voting while providing transparent, trustworthy, auditable elections. Galois has been managing the technical aspects of the E2E VIV project. In this role we have developed a comprehensive set of requirements that an E2E VIV system must satisfy, a high-level architecture for such a system, and demonstration implementations of some of its components.
Currently, the cryptography, security, auditing, and elections experts involved in the project (including the likes of Josh Beneloh, David Wagner, Dan Wallach, David Jefferson, Philip Stark, and the elections officials responsible for Los Angeles, Austin, Tallahassee, and Albuquerque) are reflecting upon these artifacts and attempting to design a new generation of E2E VIV system that can satisfy the needs of overseas, military, and disabled voters.

In this talk we will reflect upon the background and state of the E2E VIV project, focusing on the novel privacy-preserving and high-assurance aspects of the system from both algorithmic and systems points-of-view.
Abstract:

Any Coq user who has attempted a non-trivial proof has found that the process is extremely tedious. The author, after analyzing some of his own workflow in developing proofs, identified a number of areas in which the proof development process could be improved. One key finding is that developing a large proof (with many lemmas) often requires many iterations of revisions on the statement of the proof. Developing the proof script often reveals errors in the statement of the proof. Changing the statement then requires the proof to be replayed which is very tedious. We introduce a new IDE, CoqPIE that has all the functionality of Proof General or Coq IDE plus many new features to deal with workflow issues. For example, the IDE introduces tools to automatically re-play and update proof scripts. Our CoqPIE IDE consists of about 8500 lines of Python code. The tkinter library is used which connects Python to Tk/Tcl. The IDE includes its own parser for Coq syntax to implement incremental reparsing and dependency management. Other key innovations include full project management, pre-running an entire project and saving Coq output at each proof step for later reference, and many useful tactics to transform proofs.
The UI consists of a window with three views. The view on the left shows an overview of the entire project. The first level of nodes are the files in the project. The second level are definitions in the file. For proofs, additional levels exist for the tactics used in the proof. The middle view displays the source file selected in the left view. The view on the right is similar to the Coq state window in Coq IDE and Proof General. It shows the current goal and hypotheses. However, instead of showing the state at the current processing point of Coq, it shows the state just after the selected definition or proof step from the treeview at the left.
Runtime Assurance for Complex Autonomy

John Schierman (Barron Associates, Inc.)

Abstract:

For over a decade there have been important advances in control systems technologies that have enabled fast adaptation to changing environmental conditions and large unforeseen system dynamic changes. These advances in adaptive capabilities can provide significant benefit to the unmanned aircraft community, turning remotely piloted vehicles into truly autonomous unmanned aircraft systems functions. There is now wide interest in developing teams of UASs performing various cooperative operations, such as terrain mapping, search missions, high-value escort, border patrol, supply delivery, and military operations, such as surveillance and reconnaissance.

However the actual implementation of even the basic building blocks for such autonomy have still not been realized on a wide scale due to the lack of useful analysis tools in the V&V process required for flight and mission critical certification. This is mainly due to the highly advanced, complex and potentially nondeterministic nature of these new systems. For autonomous systems with no human oversight, even mission critical systems now have safety implications, requiring stringent certification.

Along with advances in design-time V&V methods, there has been considerable development of runtime assurance (RTA) systems to provide additional levels of protection during mission operations to unforeseen software errors or overall design flaws that can result in critical safety violations. This presentation will cover the latest progress from an on-going Air Force initiative that began in 2004. Our current focus has been on developing feasible frameworks for interacting RTA systems and the decision making protocols required to maintain overall system safety of the entire UAS fleet executing the current mission.

An interacting RTA system framework is a natural result of the sequential loop closure architecture of aerospace command/control systems – from (1) inner-loop attitude stabilization to (2) guidance systems directing the flight path to (3) flight management systems determining the flight path
to (4) mission planning systems generating assignment allocations and overall mission goals for the entire fleet of UAS platforms. We assume here a decentralized but cooperative command/control architecture in which the mission planning is “negotiated” by each platform’s autonomous systems through intra-fleet communications. We further assume that advanced, uncertifiable elements can exist at each of the aforementioned feedback levels, thereby requiring RTA monitoring at each level.

Our current framework defines RTA monitoring systems as assume-guarantee contract checkers, a formal methods approach from compositional reasoning. Key to this investigation is how to construct and analyze such disparate contracts with differing “languages” at each feedback level. Contracts at the inner-loop control level will involve control theoretic measures, such as norm bounds on tracking errors. However, contracts at the flight management or mission planning level may involve more discrete decision making attributes whereby it may be best to define the subsystem contracts in terms of, for example, linear temporal logic.
Inferring Contracts and Proving Properties Using Abstract Interpretation Over a Global Value Numbering

Tucker Taft (AdaCore)

Abstract:

Abstract Interpretation is widely recognized as a sound basis for doing whole program static analysis. Abstract Interpretation has classically been based on representing the values of program variables using abstract domains with appropriate widening operators to allow the set of possible abstract states at every program point to reach a fixed point. There are a number of challenges with this approach when the number of program variables is large, in that the representation of the set of possible abstract states is necessarily complex. To reduce the level of complexity in the representation of the state of a combination of related variables, we have taken an unconventional approach, based on “value numbering.” We convert the program to a “single-static-assignment” (SSA) representation and then perform a global value numbering. Subsequent analysis is performed over possible value sets associated with each value number.

The net effect of this approach is that we are able to efficiently and precisely analyze whole programs, propagating inferred contracts bottom-up through the call tree, and iterating to a fixed-point over the whole program in the case of recursion. This technology forms the basis for the commercial CodePeer static analysis tool from AdaCore, and has demonstrated advantages in scalability and precision over more conventional abstract interpretation approaches, while generating as-built documentation in the form of inferred contracts to support code review.
Bringing Roots of Trust to Reality
Adam Wick (Galois, Inc.)

Abstract:
One of the key themes in the security community over the last few years has been how to create a strong root of trust, particularly in the mobile space. After all, a strong root of trust enables a wide variety of capabilities across a number of industries. Some researchers have responded to this call with various solutions across the mobile and traditional computing worlds, while other researchers have shown how we can use these technologies in various protocols and system designs.

Yet, at the same time, mobile devices with a root of trust — strong or not — are rare. Further, if you speak to mobile device manufacturers, as we have, this rarity is not going to change any time soon. The truth is that there is no market for a strong root of trust in the mobile ecosystem, and without that market, there is no motive for an OEM to spend the time and money integrating a “useless” new technology.

Galois is currently working with the Department of Homeland Security to change this. Our thesis is that the only way to change this fact — the only way to bring mobile roots of trust into commercial relevance — is for the community to start building applications that call for a root of trust. If such apps exist, even if they use a weak root of trust, then we can, perhaps, kickstart a market.
The conference dinner will be held at the Chart House restaurant on Wednesday, May 6 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 only) 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
Armadillo’s Bar & Grill – 132 Dock Street, Annapolis, MD 21401  
Veteran American grill offering burgers & beers along with dock views & occasional live music.

Blackwall Hitch – 400 6th Street, Annapolis, MD 21403  
Upscale-casual New American restaurant featuring outside seating, an on-site pub & stylish decor.

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

Chick and Ruths Delly – 165 Main Street, Annapolis, MD 21401  
Lively landmark diner featuring greasy-spoon breakfasts & piled-high sandwiches in kitschy environs.

Cracker Barrel – 115 Blue Jay Court, Stevensville, MD 21666  
Brad’s favorite. Ask for the secret wine list!

Crush Kitchen and Wine Bar – 114 West St., Annapolis, MD 21401  
Hip wine & cocktail bar serving creative American small plates in a loungey, colorful setting.

Davis’ Pub – 400 Chester Avenue, Annapolis, MD 21403  
Featured on Diners, Drive-Ins and Dives. Try the crab pretzel!

Dock Street Bar & Grill – 136 Dock Street, Annapolis, MD 21401  
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.

Iron Rooster – 12 Market Space, Annapolis, MD 21401  
Creative all-day breakfast menu and American comfort food.

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
*Fresh, authentic Thai Cuisine in a warm contemporary environment.*

Level – A Small Plates Lounge – 69 West St., Annapolis, MD 21401
*Easygoing, wood-accented haunt offering eco-friendly New American small plates & creative cocktails.*

Metropolitan Kitchen and Lounge – 169 West St., Annapolis, MD 21401
*Casual restaurant with a broad American menu & full bar plus rooftop deck & local live music.*

O’Brien’s Oyster Bar – 113 Main St., Annapolis, MD 21401
*Imaginative seafood dishes and nouveau American cuisine. Dancing and live entertainment nightly.*

Paladar – 1905 Town Center Blvd., Annapolis, MD 21401
*Latin kitchen and rum. Katie recommends the fish tacos!*

Pusser’s Caribbean Grill – 80 Compromise St., Annapolis, MD 21401
*Waterfront location with a beautiful view.*

Rams Head Tavern – 33 West St., Annapolis, MD 21401
*An Annapolis landmark since 1989!*

Ristorante Piccola Roma – 200 Main Street, Annapolis, MD 21403
*Classic Italian favorites as well as innovative dishes.*

Sofi’s Crepes – 1 Craig Street, Annapolis, MD 21401
*Brunch crepes with sweet and savory fillings.*

Tsunami Sushi Bar and Lounge – 51 West St., Annapolis, MD 21401
*Upbeat modern Asian-fusion eatery & lounge, serving sushi, steak & seafood, with creative cocktails.*

VIDA Taco Bar – 200 Main Street, Annapolis, MD 21401
*VIDA Taco Bar offers the freshest, street food style tacos in a fun, hip, and energetic atmosphere.*

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