THE TWENTIETH ANNUAL

HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE

September 14-17, 2020

HCSS

http://cps-hcss.org

The Twentieth Annual

HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE

http://cps-hcss.org

Virtual Conference | September 14-17, 2020

WELCOME MESSAGE

On behalf of the HCSS steering committee and the NITRD CNPS Interagency Working Group, it is our great pleasure to welcome you to the 20th annual High Confidence Software and Systems (HCSS) Conference. This year is special in two ways. First, the world's unprecedented health situation has changed HCSS to a fully virtual conference rather than a physical meeting. Second, HCSS is celebrating 20 years of providing a forum for advances in high-assurance software and systems.

This year's program continues a tradition of excellence at HCSS. World-class speakers from academia, industry, and government will draw on a broad range of experiences to deliver four days of technical talks. These presentations will describe new scientific and technological foundations that will enable new generations of engineered systems for computing, communication, and control. The technologies discussed will enable the creation of high-assurance systems essential for life-, safety-, security-, and mission-critical operation.

This year's HCSS talks focus on the themes of Formal Methods at Scale, Architecture-Level Formal Methods for New and Existing Systems, and Human/Machine Cognitive Security. These themes and other topics will also be explored through technical posters at the virtual poster session.

The HCSS Conference will include two panels, first a panel allowing for refections on the 20 years of the HCSS Conference, and secondly a summary discussion relating two workshops convened in 2019 on the topic of Formal Methods at Scale.

We hope that you will find the 2020 Conference stimulating and informational. We greatly appreciate your attendance, and look forward to your participation and support at future HCSS conferences.

HCSS Co-Chairs

June Andronick, CSIRO's Data61 and UNSW
Eric Smith, Kestrel Institute

CO-CHAIRS

June Andronick
CSIRO's Data61 & UNSW

June Andronick leads the Trustworthy Systems group, world-leading in the area of verified operating systems software, known worldwide for the formal verification of the seL4 microkernel. She is a Principal Research Scientist at CSIRO's Data61, and conjoint Associate Professor at UNSW Sydney, Australia. Her main research interest is in formal verification and certification of software systems, more precisely in formal proofs of correctness and security properties of programs using interactive theorem proving, as well as concurrency reasoning, targeting interruptible and multicore systems. She was recognised in 2011 by MIT's Technology Review as one of the world's top young innovators (TR35). She previously worked in industry for the smart-card manufacturer Gemalto in Formal Methods research, where she did my PhD in collaboration with University of Paris Sud.

Eric Smith
Kestrel Institute

Dr. Eric W. Smith is the CEO of Kestrel Institute, where he does research in formal methods. He co-leads Kestrel's APT (Automated Program Transformations) project, which is developing proof-emitting program transformations for the ACL2 theorem prover. Dr. Smith also works on the Axe tool originally written for his Stanford Ph.D. thesis. Axe can lift Java bytecode and x86 binary programs into a logical representation which is then verified using Axe's high-performance rewriter and equivalence checker. Dr. Smith currently leads Kestrel projects to synthesize network protocol implementations and high-assurance monitors for machine learning algorithms. He led Kestrel's DerivationMiner effort, which used program synthesis techniques (derivations, specifications, and refinements) to construct correctness proofs of code found in large online repositories. He also led Kestrel's DARPA APAC effort to find malware in Android apps and formally prove the correctness of apps without malware.

CONFERENCE ORGANIZATION

PROGRAM CO-CHAIRS

June Andronick, CSIRO's Data61
Eric Smith, Kestrel Institute

STEERING COMMITTEE

Perry Alexander, University of Kansas
Kathleen Fisher, Tufts University
John Hatcliff, Kansas State University
John Launchbury, Galois, Inc.
Stephen Magill, MuseDev
Brad Martin, National Security Agency
Ray Richards, DARPA
William Scherlis, DARPA
Sean Weaver, National Security Agency
Matthew Wilding, Collins Aerospace

MEETING ORGANIZERS

Katie Dey, Vanderbilt University
Amy Karns, Vanderbilt University
Regan Williams, Vanderbilt University

SPONSOR AGENCY

NITRD HCSS Coordinating Group

TABLE OF CONTENTS

GENERAL INFORMATION

VIRTUAL VENUE

The 2020 HCSS Conference will be hosted on the Hopin virtual conferencing platform. Hopin is a virtual venue with multiple interactive areas that are optimized for connecting and engaging. To gain access to the virtual conference platform please register at: http://cps-hcss.org.

POSTER PRESENTATIONS

Attendees may peruse the virtual poster booths to view pre-recorded content throughout the conference. A live virtual poster session will be held from 2:00 p.m. to 3:30 p.m. Eastern on Tuesday, September 15. Stop by then to view a live demonstration and video chat with the poster presenter.

CONFERENCE MATERIALS

Conference presentations and posters will be available online at http://cps-hcss.org.

SURVEY

Please take a moment to respond to our short survey at:

https://cps-vo.org/group/hcss2020/survey

POLICIES

HCSS is committed to providing a safe and enjoyable event experience for all participants, and a welcoming environment for free discussion of ideas. We do not tolerate harassment of participants in any form during events or in any HCSS online space or social media. If you have any concerns about inappropriate behavior by any participant during the event, please contact the HCSS organizers:

File an Incident Report: https://cps-vo.org/hcss/incident-report

Email: hcss@cps-vo.org

Hopin chat: send a message to Katie Dey

Please review the complete code of conduct at: https://cps-vo.org/group/hcss/policies

PROGRAM AGENDA

PROGRAM AGENDA

MONDAY, SEPT. 14

THEME: Architecture-level Formal Methods for New and Existing Systems

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1100 - 1145	**PANEL** HCSS 20th Anniversary Reflections	*Mark Jones (Portland State University)* *John Launchbury (Galois)* *Brad Martin (NSA)* *Matt Wilding (Collins Aerospace)*	13
1145- 1215	Run-Time Assurance Architecture for Learning-Enabled Systems	*Darren Cofer (Collins Aerospace)*	15
1215- 1300	**BREAK**		
1300 - 1330	**KEYNOTE PRESENTATION** IDAS: Intent-Defined Adaptive Software ARCOS: Automated Rapid Certification of Software	*Raymond Richards (DARPA)*	17
1330- 1400	Actionable definition of Safety Design Patterns using AADLv2, ALISA and the Error Modeling Annex	*Jérôme Hugues (CMU/SEI)*	18
1400 - 1430	**BREAK**		
1430 - 1530	**KEYNOTE PRESENTATION** Take a SEAT: Security-Enhancing Architectural Transformations	*Konrad Slind (Collins Aerospace)*	20
1530 - 1600	A Verified Optimizer for Quantum Circuits	*Robert Rand (University of Chicago)*	21
1600 - 1630	Cryptographic Protocol Verification in AWS	*Adam Petcher (Amazon Web Services)*	23
1630 - 1700	**NETWORKING**		

TUESDAY, SEPT. 15

PROGRAM AGENDA

THEME: Cognitive Security

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1100 - 1200	**KEYNOTE PRESENTATION** Trustworthy AI	*Jeannette Wing* *(Columbia)*	24
1200 - 1230	Improving Computer Security by Understanding Cognitive Security	*Kimberly Feruson-Walter* *(Department of Defense)*	26
1230 - 1315	**BREAK**		
1315 - 1400	The Changing Face of Computational Propaganda: AI, Encryption, Geofencing and the Manipulation of Public Opinion	*Samuel Woolley* *(University of Texas, Austin)*	28
1400 - 1530	**POSTER & NETWORKING SESSION**		54
	A New Kind of Program Logic	*Lindsay Errington* *(Noddle)*	56
	Affix: Micro-executing Binaries to Produce Static Analysis Models	*Anwar Mamat* *(Correct Computation, Inc)*	57
	Architecture Modeling for Resource Margin Estimation	*Srini Srinivasan* *(nHansa)*	59
	Memory Bugs Classes in the NIST Bugs Framework (BF)	**Irena Bojanova* ***Carlos Galhardo* **NIST, **INMETRO*	60
	VisionGuard: Runtime Detection of Adversarial Inputs to Perception Systems	*Yiannis Kantaros* *(University of Pennsylvania)*	61

PROGRAM AGENDA

WEDNESDAY, SEPT. 16

THEME: Formal Methods at Scale

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1100 - 1200	**PANEL** FM@Scale Workshop Summary	*Patrick Lincoln (SRI)* *Brad Martin (NSA)* *William Scherlis (DARPA)*	30
1200 - 1230	Integration Challenges in Static Analysis and Verification	*Stephen Magill (MuseDev)*	32
1230 - 1315	**BREAK**		
1315 - 1400	Formal Verification of Production Distributed Protocols	*Mike Dodds (Galois)*	34
1400 - 1430	Verifying C++ at Scale	*Gregory Malecha (BedRock Systems)*	35
1430 - 1500	Analysis of the Secure Remote Password Protocol Using CPSA	*Erin Lanus (Virginia Tech)*	37
1500 - 1530	**BREAK**		
1530 - 1600	3C: Interactive Conversion of C to Checked C	*Michael Hicks (Correct Computation Inc. / University of Maryland)*	39
1600 - 1630	Adapting to demand: seL4 proofs and proof engineering practice	*Matthew Brecknell (Data61, CSIRO, Australia)*	41
1630 - 1700	**NETWORKING**		

THURSDAY, SEPT. 17

PROGRAM AGENDA

THEME: Formal Methods at Scale

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1100 - 1200	**KEYNOTE PRESENTATION** Distributed and Trustworthy Automated Reasoning	*Marijn Heule (Carnegie Mellon University)*	43
1200 - 1230	Geometric Path Enumeration Methods for Verification of ReLU Neural Networks	*Stanley Bak (Stony Brook University / Safe Sky Analytics*	44
1230 - 1315	**BREAK**		
1315 - 1345	Scaling Continuous Verification	*Joey Dodds (Galois)*	46
1345 - 1415	End-to-End Verification of Initial and Transition Properties of GR(1) Designs Generated by Salty in SPARK	*Laura Humphrey (Air Force Research Laboratory)*	47
1415 - 1445	Scalable Sound Static Analysis of Real-world C Applications using Abstract Interpretation	*Henny Sipma (Kestrel Technology)*	49
1445 - 1530	**BREAK**		
1530 - 1600	One-click Automated Reasoning in Amazon Web Services	*Andrew Gacek (Amazon Web Services)*	51
1600 - 1630	Code-Level Model Checking in the Software Development Workflow	*Daniel Schwartz-Narbonne (Amazon Web Services)*	52
1630 - 1700	**NETWORKING**		

CONFERENCE PRESENTATIONS

bold name denotes presenter

PANEL PRESENTATION
HCSS 20TH ANNIVERSARY REFLECTIONS

***Mark Jones, **John Launchbury, †Brad Martin, ††Matt Wilding**

**Portland State Univeristy, **Galois, †National Security Agency, ††Collins Aerospace*

The conference will kick off with a panel allowing for reflections on the twenty years of the HCSS conference.

Mark Jones is a professor and the current chair of the Department of Computer Science at Portland State University in Portland, Oregon. He is known, in particular, for his contributions to the functional programming language Haskell, including the design of key type system features, and the original implementation of Hugs, an early Haskell interpreter that has been widely used for teaching and research. Most recently, his work has focused on the role of programming languages and formal methods in the development, evaluation, and analysis of high-assurance, low-level software systems.

John Launchbury is the Chief Scientist at Galois, collaborating with government and industry leaders to fundamentally improve the security of software and cyber-physical systems though applied formal mathematical techniques.

Prior to rejoining Galois in 2017, John was the Director of the Information Innovation Office (I2O) at DARPA, where he led nation-scale investments in cybersecurity and artificial intelligence.

Before founding Galois in 1999, Dr. Launchbury was a full professor in Computer Science, and he is internationally recognized for his work on the analysis and semantics of functional programming languages.

John received First Class Honors in Mathematics from Oxford University in 1985. He holds a Ph.D. in Computing Science from University of Glasgow and won the British Computer Society's distinguished dissertation prize. In 2010, John was inducted as a Fellow of the Association for Computing Machinery (ACM).

CONFERENCE PRESENTATIONS

Brad Martin serves as the technical director within NSA's Laboratory for Advanced Cybersecurity Research, the U.S. government's premier cybersecurity research and design center; focused on conducting and sponsoring collaborative research in the technologies and techniques which will secure America's information systems of tomorrow. Mr. Martin has a strong history in building communities in the area of high confidence software and systems research and development, as well as having initiated research groups at NSA supporting development of supporting scientific foundations and technologies. Mr. Martin serves as Co-Chair of the Networking and Information Technology Research and Development (NITRD) Program's Computing-Enabled Networked Physical Systems (CNPS) Interagency Working Group (IWG). The CNPS IWG coordinates Federal R&D to advance and assure information technology-enabled systems that integrate the cyber/information, physical, and human elements. Additionally, Mr. Martin previously served as the Chair of the Special Cyber Operations Research and Engineering (SCORE) Subcommittee, a Subcommittee of the NSTC Committee on Homeland & National Security. The SCORE Subcommittee is focused on enhancing coordination and collaboration across the cyber research community, and specifically scoped for science and technology for national security needs in cyber.

Matthew Wilding is an Associate Director at Collins Aerospace. Dr. Wilding received a PhD in Computer Sciences from The University of Texas at Austin and joined Collins Aerospace in 1996. He has worked on many high confidence efforts, such as the machine-checked formal methods analysis of the AAMP7 microprocessor critical to several Collins information assurance products. Dr. Wilding led the digital vision systems research group from 2011-2016, which developed the Integrated Digital Vision System (IDVS) to enhance soldier situational awareness. He currently manages the Trusted Systems research group, which collaborates with corporate product areas and government research sponsors to develop and apply automated verification methods for complex embedded systems.

RUN-TIME ASSURANCE ARCHITECTURE FOR LEARNING-ENABLED SYSTEMS

***Darren Cofer,** *Isaac Amundson, *Ram Sattigeri, *Arjun Passi, *Chris Boggs, **Eric Smith, **Limei Gilham, †Taejoon Byun, †Sanjai Rayadurgam

**Collins Aerospace, **Kestrel Institute, †University of Minnesota*

There has been much publicity surrounding the use of machine learning technologies in self-driving cars and the challenges this presents for guaranteeing safety. These technologies are also being investigated for use in manned and unmanned aircraft. However, systems including "learning-enabled components" (LECs) and their software implementations are not amenable to verification and certification using current methods. This limits the functionality that can realistically be fielded, and essentially precludes use of these technologies in safety-critical aerospace applications.

Our team is developing new technologies for analysis, testing, and architectural mitigation, with the goal of enabling autonomous systems containing LECs to be safely deployed in critical environments. We have produced a demonstration of a run-time assurance architecture based on a neural network aircraft taxiing application that shows how several advanced technologies could be used to ensure safe operation. The demonstration system includes:

- Safety architecture based on the ASTM F3269-17 standard for bounded behavior of complex systems
- AADL architecture model with verification using AGREE
- Architecture-based assurance case using Resolute
- Diverse run-time monitors of system safety
- Formal synthesis of critical high-assurance components

The enhanced system demonstrates the ability of the run-time assurance architecture to maintain system safety in the presence of defects in the underlying LEC.

CONFERENCE PRESENTATIONS

Darren Cofer is a Fellow in the Trusted Systems group at Collins Aerospace. He earned his PhD in Electrical and Computer Engineering from The University of Texas at Austin.

His principal area of expertise is developing and applying advanced analysis methods and tools for verification and certification of high-integrity systems. His background includes work with formal methods for system and software analysis, the design of real-time embedded systems for safety-critical applications, and the development of nuclear propulsion systems in the U.S. Navy.

He has served as principal investigator on government-sponsored research programs with NASA, NSA, AFRL, and DARPA, developing and using formal methods for verification of safety and security properties. He is currently the principal investigator for Collins teams working on DARPA's Cyber Assured Systems Engineering (CASE) and Assured Autonomy programs.

Dr. Cofer served on RTCA committee SC-205 developing new certification guidance for airborne software (DO-178C) and was one of the developers of the Formal Methods Supplement (DO-333). He is a member of SAE committee G-34 on Artificial Intelligence in Aviation, the Aerospace Control and Guidance Systems Committee (ACGSC), and a senior member of the IEEE.

KEYNOTE PRESENTATION

IDAS: INTENT-DEFINED ADAPTIVE SOFTWARE ARCOS: AUTOMATED RAPID CERTIFICATION OF SOFTWARE

Raymond Richards
DARPA

ACTIONABLE DEFINITION OF SAFETY DESIGN PATTERNS USING AADLV2, ALISA AND THE ERROR MODELING ANNEX

Jérôme Hugues
Carnegie Mellon University, Software Engineering Institute (CMU/SEI)

Demonstrating safety of critical systems is achieved by the careful examination of evidences built from software, hardware, functional, and non-functional properties and the architecture that combines them to form the overall system. Formal methods (model checking, theorem proving) provide evidences that are later combined to form a system's assurance case. Error Taxonomies provide guideline to evaluate errors or faults that may affect a system. Design patterns provide reusable solutions to recurring engineering problem to guide the system architect.

Surveys like [Preschern et al] provides both a definition of safety-related design patterns, and the architectural design decisions they imply. Surprisingly, deriving a correct instantiation of these patterns for an actual system, and the associated verification plan is an open question. Patterns are defined in a very abstract way, that must be adjusted to a specific project. They lack any actionable definitions that can be processed, for both their architectural core concepts, and the verification plan they imply.

The AADL group at CMU/SEI is currently conducting a study to model these design patterns using the AADL, and later relate them to actual system's architecture, but also to a verification plan. AADL provides a modeling framework for describing the architecture of hardware and platform resources, software components, and flexible allocation software components to resources. Through its annex languages and tool plug-in extensibility mechanisms, it also provides a variety of architecture analyses including hazard analysis, schedulability analysis, dependence analysis. In addition, ALISA supports the definition of assurance plan that relates an architectural description to a set of verification methods connected to formal analysis. In this talk, we will illustrate how to leverage AADL and its ecosystem to capture Safety design patterns as a library of model elements; capture for each pattern the corresponding abstract verification plan they presume; and then apply them to specific system instances. We will discuss different usage of AADL to ensure correct traceability between a design pattern

definition and its instantiation, either through extension and refinement; or through the preservation of design patterns structural invariants based on a graph representation of the pattern. Hence, we illustrate how these models can be used as actionable definition of patterns.

Reference:

[1] Preschern C., Kajtazovic N., Kreiner C. (2019) Safety Architecture Pattern System with Security Aspects. In: Noble J., Johnson R., Zdun U., Wallingford E. (eds) Transactions on Pattern Languages of Programming IV. Lecture Notes in Computer Science, vol 10600. Springer, Cham

Jérôme Hugues is Senior Researcher at the Software Enigneering Institute on the Assuring Cyber-Physical Systems team. He holds a PhD (2005) and an engineering degree (2002) from Telecom ParisTech. His research interests focus on design of software-based real- time and embedded systems and tools to support it. He is a member of the SAE AS-2C committee working on the AADL since 2005. Prior to joining the CMU/SEI, he was professor at the Department of Engineering of Complex Systems of the Institute for Space and Aeronautics Engineering (ISAE), in charge of teaching curriculum on systems engineering, safety-critical systems and real-time systems. He contributes to the OSATE, Ocarina and TASTE projects AADL toolchains.

KEYNOTE PRESENTATION

TAKE A SEAT: SECURITY-ENHANCING ARCHITECTURAL TRANSFORMATIONS

***Konrad Slind,** *David Hardin, *Thomas Logan, *Junaid Babar,
**Eric Mercer, †Johannes Pohjola, *Darren Cofer, *Isaac Amundsen
***Brigham Young University, *Collins Aerospace, †Data61*

System architecture models in a language such as AADL provide a high-level setting in which existing implementations, new design features, high-level requirements, and verifications can be combined. We have recently been developing selected architecture-to-architecture transformations as a way to enhance the security of a system. The transformations are formally specified at a high level and mapped to implementations by a sequence of correctness-preserving (and deductively justified) compilation steps. We will present details of two transformations: one that creates efficient network message filters from regular expressions, and one that creates runtime monitors from temporal logic specifications. For both transformations, we illustrate an automated, end-to-end, verification path that formally connects top-level component specifications with the binary behavior of the newly introduced security gadgets. Since these gadgets are expected to run in an embedded system, further properties, such as execution time and liveness, are required, and we will discuss how our framework can prove those as well. Time permitting, we will also discuss aspects of how this formal toolchain is incorporated into a build system which maps AADL architecture models to system images in seL4 and Linux.

Dr. Konrad Slind (PhD TU Munich) is a Senior Industrial Logician in the Trusted Systems Group of Collins Aerospace. He has been at Collins Aerospace for ten years; previous stints were at University of Utah School of Computing (faculty), Cambridge University Computer Lab, and Bell Labs. Slind's main area of research is the design, implementation, and application of interactive theorem provers. He has published on a variety of verification topics, including hardware verification, compiler verification, and block cipher verification. At Collins, he is currently working on the DARPA CASE project.

A VERIFIED OPTIMIZER FOR QUANTUM CIRCUITS

***Robert Rand,** **Michael Hicks, **Kesha Hietala,
**Shih-Han Hung, *Xiaodi Wu
**University of Chicago, **University of Maryland*

Programming quantum computers will be challenging, at least in the near term. Qubits will be scarce, and gate pipelines will need to be short to prevent decoherence. Fortunately, optimizing compilers can transform a source algorithm to work with fewer resources. Unfortunately, there is a risk that such compilers will operate incorrectly, potentially in a way that is hard to detect; the result could be wrong answers or even security vulnerabilities. In fact, bugs have been found in several state of the art compilers, including IBM's popular Qiskit compiler.

We present voqc (Figure 1), the rst compiler for quantum circuits fully veried to be correct. Quantum circuits are expressed as programs in a simple, low-level language called sqir, which is deeply embedded in the language of the Coq proof assistant. Optimizations and other transformations are expressed as Coq functions, which are proved correct with respect to a semantics of sqir programs. These functions are extracted OCaml by a standard process and then compiled to the nal compiler executable. We evaluate voqc's veried optimizations by running it on a series of benchmarks (written in the standard OpenQASM format). voqc performs comparably to industrial-strength compilers (while being provably bug-free): voqc's optimizations reduce total gate counts on average by 18.4% on a benchmark of 29 circuit programs compared to a 10.7% reduction when using IBM's Qiskit compiler and a 11.2% reduction when using CQC's t|ket> compiler.

VOQC is freely available at https://github.com/inQWIRE/SQIR

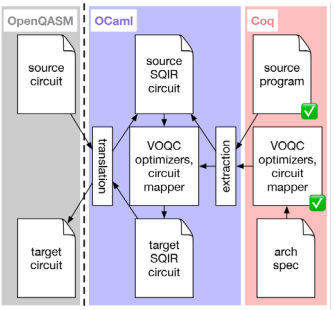

Figure 1: VOQC architecture

Robert Rand is an assistant professor at the University of Chicago with a research specialty in quantum programming. Rand applies techniques from programming languages and formal verification to quantum computation, creating tools for writing, testing, and running reliable software on the quantum computers of today and tomorrow. He co-developed QWIRE, a quantum circuit language, and VOQC, a verified optimizing compiler for quantum programs, and incorporates elements of both in his online textbook, Verified Quantum Computing.

CRYPTOGRAPHIC PROTOCOL VERIFICATION IN AWS

Adam Petcher
Amazon Web Services

In this talk, I will describe how Amazon Web Services verifies the cryptographic protocols that are used to secure its cloud platform. This verification is accomplished through the mechanization of cryptographic security proofs in the computational model. In some cases, we mechanize proofs in existing tools, such as EasyCrypt. In order to reason about complex protocols, we developed Quivela, which is capable of proving asymptotic indistinguishability between real and ideal functionalities in a universally composable setting. Verified protocols include the domain management protocol used by the AWS Key Management Service, and the compromise-resilient signature protocol that authenticates all AWS requests.

Protocol verification is a useful component of design/architecture verification at AWS, and it allows us to identify issues in existing designs, and explore alternatives in new designs. By producing verified protocol models, we also get closer to the eventual goal of a fully verified design and implementation, in which the model serves as a functional specification for the implementation.

Adam Petcher is an applied scientist at Amazon Web Services who specializes in cryptography and formal verification. His current work is in the area of machine-checked cryptographic proofs and the formal verification of cryptographic implementations. Prior to AWS, Adam worked on applied cryptography at Oracle and MIT Lincoln Laboratory.

KEYNOTE PRESENTATION
TRUSTWORTHY AI

Jeannette Wing
Columbia University

Recent years have seen an astounding growth in deployment of AI systems in critical domains such as autonomous vehicles, criminal justice, healthcare, hiring, housing, human resource management, law enforcement, and public safety, where decisions taken by AI agents directly impact human lives. Consequently, there is an increasing concern if these decisions can be trusted to be correct, reliable, fair, and safe, especially under adversarial attacks. How then can we deliver on the promise of the benefits of AI but address these scenarios that have life-critical consequences for people and society? In short, how can we achieve trustworthy AI?

Under the umbrella of trustworthy computing, there is a long-established framework employing formal methods and verification techniques for ensuring trust properties like reliability, security, and privacy of traditional software and hardware systems. Just as for trustworthy computing, formal verification could be an effective approach for building trust in AI-based systems. However, the set of properties needs to be extended beyond reliability, security, and privacy to include fairness, robustness, probabilistic accuracy under uncertainty, and other properties yet to be identified and defined. Further, there is a need for new property specifications and verification techniques to handle new kinds of artifacts, e.g., data distributions, probabilistic programs, and machine learning based models that may learn and adapt automatically over time. This talk will pose a new research agenda, from a formal methods perspective, for us to increase trust in AI systems.

Jeannette Wing is the Avanessians Director of the Data Science Institute and Professor of Computer Science at Columbia University.

From 2013 to 2017, she was a Corporate Vice President of Microsoft Research. She is Adjunct Professor of Computer Science at Carnegie Mellon where she twice served as the Head of the Computer Science Department and had been on the faculty since 1985. From 2007-2010 she was the Assistant Director of the Computer and Information Science and Engineering Directorate at the National Science Foundation. She received her S.B., S.M., and Ph.D. degrees in Computer Science, all from the Massachusetts Institute of Technology.

Professor Wing's general research interests are in the areas of trustworthy computing, specification and verification, concurrent and distributed systems, programming languages, and software engineering.

Her current interests are in the foundations of security and privacy, with a new focus on trustworthy AI. She was or is on the editorial board of twelve journals, including the Journal of the ACM and Communications of the ACM.

Professor Wing is known for her work on linearizability, behavioral subtyping, attack graphs, and privacy-compliance checkers. Her 2006 seminal essay, titled "Computational Thinking", is credited with helping to establish the centrality of computer science to problem-solving in fields where previously it had not been embraced.

She is currently a member of: the National Library of Medicine Blue Ribbon Panel; the Science, Engineering, and Technology Advisory Committee for the American Academy for Arts and Sciences; the Board of Trustees for the Institute of Pure and Applied Mathematics; the Advisory Board for the Association for Women in Mathematics; and the Alibaba DAMO Technical Advisory Board. She has been chair and/or a member of many other academic, government, and industry advisory boards. She received the CRA Distinguished Service Award in 2011 and the ACM Distinguished Service Award in 2014. She is a Fellow of the American Academy of Arts and Sciences, American Association for the Advancement of Science, the Association for Computing Machinery (ACM), and the Institute of Electrical and Electronic Engineers (IEEE).

IMPROVING COMPUTER SECURITY BY UNDERSTANDING COGNITIVE SECURITY

Kimberly Ferguson-Walter
National Security Agency's Laboratory for Advanced Cybersecurity Research

Abstract:

Oppositional Human Factors are a new way to apply well-known research on decision-making biases and human attention allocation to disrupt potential cyber attackers and provide much-needed asymmetric benefits to the defender. There has been significant research on how we perform cyber defense tasks and how we should present information to operators, cyber defenders, and analysts to make them more efficient and more effective. Inverting human factors can aid in cyber defense by flipping well-known guidelines and using them to degrade and disrupt the performance of a cyber attacker. Defensive cyber deception is one well-research technique used to induce some of these cognitive biases in cyber attackers. We designed the Tularosa Study, to better understand how defensive deception, both cyber and psychological, affect cyber attack behavior. Over 130 red teamers participated in a network penetration test over two days in which we controlled both the presence of and explicit mention of deceptive defensive techniques. To our knowledge, this represents the largest study of its kind ever conducted on a skilled red team population. We present results supporting a new ?finding that the combination of the presence of deception and the true information that deception is present has the greatest effect? on cyber attackers, when compared to a control condition in which no deception was used.

Dr. Kimberly Ferguson-Walter is a Senior Research Scientist with the National Security Agency's Laboratory for Advanced Cybersecurity Research. She earned a BS in Information and Computer Science from the University of California Irvine, cum laude, with a specialization in artificial intelligence and her MS and PhD in Computer Science from the University of Massachusetts Amherst. Her research interests are focused on the intersection of computer security, artificial intelligence, and human behavior. In addition to cyber security, her research background includes reinforcement learning, transfer learning, representation learning, and intelligent tutoring systems. She has been focused on adaptive cybersecurity at the NSA for the past ten years and is the lead for the Research Directorate's deception for cyber-defense effort. She is currently on joint-duty assignment to the Naval Information Warfare Center Pacific to perform collaborative research and facilitate strategic alignment and technology transfers and acts as an advisor to the Navy Science & Technology Advisor Council on matters involving Cyber and Autonomy. She has organized international workshops on cyber deception, autonomous cyber operations, and cognitive security. Dr Ferguson-Walter is a founding member of the Cybersecurity Technical Group of the Human Factors and Ergonomics Society (HFES) and co-chairs a mini-track at the Hawaiian International Conference on System Science (HICSS) on Cyber Deception and Cyber Psychology for Defense.

THE CHANGING FACE OF COMPUTATIONAL PROPAGANDA: AI, ENCRYPTION, GEOFENCING AND THE MANIPULATION OF PUBLIC OPINION

Samuel Woolley
University of Texas, Austin

There is a growing body of research on the use of social media and other digital communication tools in attempts to manipulate both human and computational systems. But how is computational propaganda, the use of automation and algorithms in online attempts to alter public opinion, changing? How can we best study these changes? This talk explores the rise of AI chatbot, rather than simple social bots, in transnational political information operations. It also discusses two new forms of computational propaganda: "geo-propaganda" and "encrypted-propaganda". The former is focused on how geo-location data, alongside other forms of data, is being used to create sophisticated disinformation and propaganda campaigns. The latter looks into the differences between manipulative content on encrypted messaging applications versus that on public social media platforms. Dr. Woolley will access ongoing research from the Propaganda Research Team at UT Austin's Center for Media Engagement to address these, and other, pressing questions.

Dr. Samuel C. Woolley is a writer and researcher focused on how emerging media technologies are leveraged for both freedom and control. His new book, The Reality Game: How the Next Wave of Technology Will Break the Truth (PublicAffairs), explores how tools from artificial intelligence to virtual reality are being used in efforts to manipulate public opinion and discusses how society can respond. Dr. Woolley is an assistant professor in the School of Journalism and the School of Information (by courtesy) at the University of Texas (UT) at Austin. He is the program director of the propaganda research lab at UT's Center for Media Engagement and director of disinformation research for the UT "Good Systems" grand challenge–a university wide project exploring ethical AI design. He is a research affiliate at the Project on Democracy and the Internet at Stanford University. He is the co-editor, with Dr. Philip N. Howard, of the book "Computational Propaganda". His research has been published by a number of academic venues including The Journal of Information, Technology, and Politics, The International Journal of Communication, and The Handbook of Media, Conflict, and Security. His writings on tech, propaganda, and policy have been published by the National Endowment for Democracy, the Brookings Institution, the Stanford Hoover Institution, USAID, and the German Marshall Fund. He is a regular contributor to Wired, the MIT Technology Review, Slate, and a number of other publications. His research has been featured in the New York Times, the Wall Street Journal, and the Financial Times. He is the former director of research of the Computational Propaganda Project at the University of Oxford and the Founding Director of the Digital Intelligence Lab at the Institute for the Future in Palo Alto, CA. He is a former fellow at the Anti-Defamation League (ADL), the German Marshall Fund of the United States, Google Jigsaw, the Tech Policy Lab, and the Center for Media, Data and Society at Central European University. He has past academic affiliations with CITRIS at UC Berkeley and the Oxford Internet Institute at the University of Oxford. His PhD is from the University of Washington in Seattle. He tweets from @samuelwoolley.

PANEL PRESENTATION
FM@SCALE WORKSHOP SUMMARY

***Patrick Lincoln, **Brad Martin, †William Scherlis**
**SRI, **National Security Agency, †DARPA*

Two workshops were convened in 2019 on the topic of Formal Methods at Scale. Participants from U.S. industry, government, and academia gathered to discuss recent advances in the application of formal methods at scale and prospects for the future. The workshops showcased excitement in the community regarding the advances in formal methods technology, the scale of existing applications, and potential for a new and broader scope for formal methods applications. Specific topics discussed included improvements in tools, practices, and training and characteristics of existing and emerging applications.

Patrick Lincoln, Ph.D., is Vice President of Information and Computing Sciences, and director of the Computer Science Laboratory (CSL) at SRI International. Lincoln leads research in the fields of formal methods, computer security and privacy, computational biology, scalable distributed systems, and collaborative interfaces. He has led multidisciplinary groups conducting high-impact research projects in symbolic systems biology, scalable anomaly detection, exquisitely sensitive biosensor systems, strategic reasoning and game theory, and privacy-preserving data sharing. He has published dozens of influential papers, holds several patents, has served on scientific advisory boards for private and publicly held companies, nonprofits, and government agencies and departments. Lincoln holds a Ph.D. in computer science from Stanford University and a B.Sc. in computer science from MIT. He has previously held positions at MCC, Los Alamos National Laboratory, and ETA Systems. Patrick was named an SRI Fellow in 2005.

Brad Martin serves as the technical director within NSA's Laboratory for Advanced Cybersecurity Research, the U.S. government's premier cybersecurity research and design center; focused on conducting and sponsoring collaborative research in the technologies and techniques which will secure America's information systems of tomorrow. Mr. Martin has a strong history in building communities in the area of high confidence software and systems research and development, as well as having initiated research groups at NSA supporting development of supporting scientific foundations and technologies. Mr. Martin serves as Co-Chair of the Networking and Information Technology Research and Development

(NITRD) Program's Computing-Enabled Networked Physical Systems (CNPS) Interagency Working Group (IWG). The CNPS IWG coordinates Federal R&D to advance and assure information technology-enabled systems that integrate the cyber/information, physical, and human elements. Additionally, Mr. Martin previously served as the Chair of the Special Cyber Operations Research and Engineering (SCORE) Subcommittee, a Subcommittee of the NSTC Committee on Homeland & National Security. The SCORE Subcommittee is focused on enhancing coordination and collaboration across the cyber research community, and specifically scoped for science and technology for national security needs in cyber.

Dr. William Scherlis assumed the role of office director for DARPA's Information Innovation Office (I2O) in September 2019. In this role he leads program managers in the development of programs, technologies, and capabilities to ensure information advantage for the United States and its allies, and coordinates this work across the Department of Defense and U.S. government.

Scherlis joined DARPA from Carnegie Mellon University (CMU), where he is a professor of computer science. He served for 12 years as director of CMU's Institute for Software Research (ISR), overseeing research and educational programs related to software development, cybersecurity, privacy engineering, Internet of Things, network analysis, mobility, systems assurance, and other topics. During 2012 and early 2013 he was the acting chief technology officer for the Software Engineering Institute, a Department of Defense FFRDC at CMU.

Earlier in his career, Scherlis served as a program manager and later in the Senior Executive Service at DARPA, developing programs in areas such as software technology, computer security, and information infrastructure. At DARPA, he also participated in the initiation of the High Performance Computing and Communications (HPCC) program (now NITRD) and in defining the concept for CERT-like security organizations, hundreds of which now operate in more than 90 countries.

Scherlis has led multiple national studies including the National Research Council study committee that produced the report "Critical Code: Software Producibility for Defense" in 2010. He also served multiple terms as a member of DARPA's Information Science and Technology Study Group. He has been an advisor to major technology firms, defense companies, and venture investors, and has served as program chair for a number of technical conferences including the ACM Foundations of Software Engineering Symposium and the ACM Symposium on Partial Evaluation and Program Manipulation. He is a fellow of the IEEE and a Lifetime National Associate of the National Academy of Sciences.

Scherlis joined the CMU faculty after completing an undergraduate degree in applied mathematics at Harvard University, a year in the Department of Artificial Intelligence at the University of Edinburgh as a John Knox Fellow, and a doctorate program in computer science at Stanford University. His personal research relates to software assurance, cybersecurity, software analysis, and assured safe concurrency.

INTEGRATION CHALLENGES IN STATIC ANALYSIS AND VERIFICATION

Stephen Magill
MuseDev

What would it take to reach a point where 80% of developers at DoD Primes and Fortune 500 companies are using formal methods based tools? Certainly tool capabilities, efficiency, and usability play an important role here. But scaling formal methods is not just about scaling analysis techniques to handle more code or more complex specifications, but also about developing the infrastructure to allow analysis tools to deliver value in larger, more complex organizations, and ensuring that analysis processes and workflows can handle systems that evolve over time.

Solving these integration challenges requires addressing technology gaps and supporting organizational and communication patterns that are largely orthogonal to tool-specific scalability concerns. In this talk, we discuss the following question: Can we develop a platform that solves these integration challenges in a way that 1) is tool agnostic (thus benefiting all tools) and 2) supports a variety of program analysis techniques, from bug-finding to sound static analysis and even extending to verification tools?

We will describe the landscape of integration challenges using concrete examples from industry collected from over 30 discussions with developers and multiple tool deployment engagements.

We will present affirmative results showing 1) that verification tools can be run in Continuous Integration / Continuous Deployment (CI / CD) workflows and provide continuing value to developers, 2) that certain tool design principles allow a common integration infrastructure to be leveraged by a variety of tools, and 3) that with the right tool support ordinary developers can evolve specifications over time. In addition to these encouraging results, we will describe remaining challenges and highlight open problems (of which there are many). In so doing, we hope to inspire further research in this area and shed light on important aspects of usability that impact adoption but that are largely invisible to formal methods researchers.

This talk covers work on:

- Integration of verification into CI to support both formal correctness proofs and formal methods based regression testing.
- A plugin architecture for formal methods based static analysis tools that supports integration with build systems, code review systems, and developer feedback channels.
- Automated update of information flow specifications to support continuous analysis of evolving codebases.

While the topic of this talk is broader than these specific solutions, these concrete examples will help ground the discussion of what can otherwise be an amorphous set of concerns, and will serve to highlight the open problems that still exist.

Dr. Stephen Magill has spent over 15 years advancing the state of software security and privacy via both basic research and technology transition efforts. Stephen currently serves as CEO of Muse Dev, a company focused on bringing advanced static analysis and verification capabilities to developers. Stephen has led several large research and development projects, including serving as Principal Investigator on a number of DARPA programs, directing research engagements with Amazon Web Services, and leading research for the 2019 State of the Software Supply Chain report. Stephen also serves on the University of Tulsa Industry Advisory Board and numerous program committees and funding panels. He has a Ph.D. in Computer Science from Carnegie Mellon University, and his work has been widely published.

FORMAL VERIFICATION OF PRODUCTION DISTRIBUTED PROTOCOLS

Mike Dodds, Giuliano Losa
Galois

Byzantine fault-tolerant (BFT) protocols underlie many highly scalable distributed services. For example, distributed databases and blockchain applications both use BFT protocols to ensure that, despite the absence of any central authority, all participants agree on the global state of the database or blockchain. Despite their importance, BFT protocols are notoriously hard to get right because the combination of arbitrary network timing, crashes, and malicious attacks makes for a large number of scenarios to consider.

In this talk we will describe a formal verification methodology suitable for verifying real-world BFT protocols. We will illustrate this methodology with a recent proof of the safety and liveness of the Stellar Consensus Protocol. Working with the Stellar team, we formalized their protocol and identified a previously unknown weakness affecting its liveness properties. Our approach therefore appears to be useful in practice when constructing highly reliable distributed applications that are intended to scale.

Our approach is based on the Ivy verifier developed by McMillan and others. In general, BFT protocols cannot be verified in a push-button manner, but interactive proof is challenging because of the unpredictability of automated solvers. To curb this problem, Ivy restricts the user to proof obligations that fall within a known and well-supported decidable logic. This increases proof-engineer productivity by ensuring that the solver rapidly either confirms the validity of the proof or provides a concrete counter-example.

Ivy can only be applied to systems that can be modelled within its restricted core logic. Surprisingly, we have shown that Ivy is expressive enough to model many intricate distributed protocols, including Stellar. This leads to more rapid verification and proofs that are much more compact than the state-of-the-art.

Dr. Mike Dodds is a Principal Scientist at Galois, Inc. His research focuses on industrial applications of formal methods. He helps lead Galois' collaboration with Amazon Web Services and Galois' SPARTA project, part of DARPA SafeDocs. Dr. Dodds joined Galois from the University of York UK, where his research group focused on automated reasoning and concurrency verification. Dr Dodds holds a PhD and Masters of Engineering from the University of York.

VERIFYING C++ AT SCALE

Gregory Malecha
BedRock Systems

At BedRock Systems, we are developing and verifying a full virtualization stack from micro-hypervisor to user-land components developed on top of it using a formal-methods-first approach centered around modularity and automation. This ambitious task requires scaling in both the size and the complexity of the code. Formal-methods-first means that our Formal Methods team is working hand-in-hand with our systems developers to specify (and subsequently verify) the code. We have found that this process leads to more modular code that is easier to both specify and verify.

Applying formal methods to mainstream languages is crucial to expanding its adoption. Our verification infrastructure is built around modern C++ and leverages the clang compiler toolchain, allowing it to integrate with existing C++ code. Supporting C++ allows engineers to verify the code they write in a language that they are already familiar with, rather than forcing them to learn an entirely new, and often painfully minimalistic, programming language.

To support a modularity-first mentality, our tooling embraces separation logic, a logic built around disjointness of resources. Two crucial features of separation logic stand out as being particularly important for scaling verification.

First, separation logic's separate-by-default "mentality" encourages thinking about modularity upfront leading to more verifiable code. In fact, we have found that many well-known patterns and anti-patterns can be justified by considering the specifications that hold on them. This lends credibility to the formal methods point of view when engaging with engineers that are new to formal methods.

Second, separation logic's expressivity allows us to apply it pervasively across all levels of our stack. The exact same principles for the user-land C++ code also apply to the kernel with page tables and embedded assembly. In the future, this will enable us to integrate with other languages, e.g. Rust or Java, that developers may wish to use on top of our stack.

While crucial, modularity is not a panacea for scale; automation is also essential. This is especially true in low-level languages such as C++ where code is often more explicit than it would be in higher-level languages. In conjunction with our verification efforts, we are building automation to reason about the more mundane aspects of verification (e.g. arithmetic reasoning) and have successfully used it to automatically verify some simple functions.

At the scale of our ambitions (which include weak memory concurrency), however, fully automated proofs are beyond the state of the art. To overcome this limitation, our tooling allows users to reason interactively about features that have not yet been automated such as

loop invariants and complex concurrency. However, our automation is also highly customizable and we have already demonstrated some simple and reusable concurrent abstractions for automated proofs of concurrent code.

Gregory Malecha received his PhD in 2015 under the direction of Greg Morrisett and Adam Chlipala focusing on automation for program verification especially for separation logics. During his dissertation he worked on applying formal verification technology to verify infrastructure software such as a webserver and a simple database management system. After his degree, he worked on the verification of cyber-physical systems in the VeriDrone project with Sorin Lerner, proving safety properties in temporal logic. After a brief hiatus doing operations research and data science at Target Corp, he returned to the verification community and is currently developing verification technology and bringing academic insights on verification to the development of a full hypervisor stack as the director of formal methods at BedRock Systems.

ANALYSIS OF THE SECURE REMOTE PASSWORD PROTOCOL USING CPSA

[§]Erin Lanus, [††]Alan T. Sherman, *Moses Liskov, **Edward Zieglar, [††]Richard Chang, [††]Enis Golaszewski, [††]Ryan Wnuk-Fink, [††]Cyrus Jian Bonyadi, [††]Mario Yaksetig, [†]Ian Blumenfeld

**The MITRE Corporation, **National Security Agency, [†]Two Six Labs, [††]University of Maryland, Baltimore County, [§]Virginia Tech*

We analyze the *Secure Remote Password (SRP)* protocol for structural weaknesses using the *Cryptographic Protocol Shapes Analyzer (CPSA)* in the rst formal analysis of SRP (specically, Version 3).

SRP is a widely deployed *Password Authenticated Key Exchange (PAKE)* protocol used in 1Password, iCloud Keychain, and other products. As with many PAKE protocols, the two participants (e.g., client and server) use knowledge of a pre-shared password to authenticate each other and establish a session key. SRP aims to resist dictionary attacks, not store plaintext-equivalent passwords on the server, avoid patent infringement, and avoid export controls by not using encryption. Formal analysis of SRP is challenging in part because existing tools provide no simple way to reason about its use of the mathematical expression "$v + g^b \bmod q$".

Modelling $v + g^b$ as encryption, we complete an exhaustive study of all possible execution sequences of SRP. Ignoring possible algebraic attacks, this analysis detects no major structural weakness, and in particular no leakage of any secrets. We do uncover one notable weakness of SRP, which follows from its design constraints. It is possible for a malicious server to fake an authentication session with the client, without the client's knowledge or participation. This action might facilitate an escalation of privilege attack, if the client has higher privileges than does the server. We conceived of this attack before we used CPSA and confirmed it by generating corresponding execution shapes using CPSA.

Keywords. Cryptographic protocols, cryptography, Cryptographic Protocol Shapes Analyzer (CPSA), formal methods, PAKE protocols, protocol analysis, Secure Remote Protocol (SRP), structural weaknesses, UMBC Protocol Analysis Lab (PAL).

Acknowledgements:

We are grateful to Edward Zieglar (NSA) and Moses Liskov (MITRE) for help with using CPSA. Thanks also to these individuals, John Ramsdell (MITRE), other participants at the Protocol eXchange, and Ian Blumenfeld for fruitful interactions.

This research was supported in part by the U.S. Department of Defense under CySP Capacity grants H98230-17-1-0387 and H98230-18-1-0321. Sherman, Oliva, Golaszewski, Wnuk-Fink, Cyrus Jian Bonyadi, and the UMBC Cyber Defense Lab were supported also in part by the National Science Foundation under SFS grant DGE-1753681.

Selected references: [Wu98, Wu00, Wu02]

References:

[Wu98] ThomasWu. The Secure Remote Password Protocol. In Proceedings of the Internet Society on Network and Distributed System Security, 1998.

[Wu00] Thomas Wu. The SRP Authentication and Key Exchange System, RFC 2945, September 2000.

[Wu02] Thomas Wu. SRP-6: Improvements and Renements to the Secure Remote Pass- word Protocol, October 2002.

Dr. Erin Lanus is a Research Assistant Professor at the Hume Center for National Security and Technology at Virginia Tech. She has a Ph.D. in Computer Science with a concentration in cybersecurity from Arizona State University. Her experience includes work as a Research Fellow at University of Maryland Baltimore County and as a High Confidence Software and Systems Researcher with the Department of Defense. Her current research interests are software and combinatorial interaction testing, machine learning in cybersecurity, and artificial intelligence assurance.

3C: INTERACTIVE CONVERSION OF C TO CHECKED C

***Michael W. Hicks,** **Aravind Machiry, *Ray Chen, *Hasan Touma, *Aaron Eline

**Correct Computation Inc., **UC Santa Barbara*

Checked C[1] is a freely available, backward compatible extension to C that aims to support spatial memory safety. Checked C defines annotations for pointer types that its compiler uses to verify, either statically or dynamically, that pointer accesses are safe. Such safety assurance is useful in itself (e.g., it precludes buffer overflows) and also sets a useful foundation for subsequent static analyzers.

To streamline the process of upgrading a legacy C program to Checked C, we have been developing a tool, **3C,** to automatically infer Checked C annotations. **3C** is designed to be interactive, in the sense that a developer will use it repeatedly on the same codebase. As shown in Figure 1, **3C** performs a wholeprogram, constraint-based static analysis to infer safe-pointer annotations. The developer can make changes to the resulting code, e.g., to correct mistakes or remove anti-patterns (e.g., unsafe casts) or to add missing annotations (e.g., due to hard-to-discover loop invariants), and then re-run **3C** to perform further work.

To support this workflow, **3C**'s static analysis has two novel elements. First, it *localizes* its determination of when pointers can be deemed safe, and expresses that at the function boundary. If a function `void foo(int *x) { ... }` uses its parameter x safely, then it can be annotated as safe in the definition, even if a caller passes in an unsafe argument, e.g., `foo((int *)5)`--we deem this a problem with the caller, not with `foo`. Conversely, if `foo` uses `x` unsafely then `x` is not annotated (as such, a caller `foo(y)` would have its argument y deemed unsafe). 3C can point out the "root cause" for a safety problem, which the programmer can fix prior to re-running 3C.

Second, **3C** is able to analyze code that is a mix of legacy and Checked C code, even when that mix is self-inconsistent, e.g., because it assigns a legacy (unsafe) pointer to a safe (annotated) one. With the experience of converting 100s of KLoC, we find that 3C's interactive approach speeds up the overall process of converting code because leverages the particular strengths of the human and machine components of the conversion process. Our talk will describe our approach, our experience using it, and the challenges we are addressing in ongoing work. **3C** is part of the public release of Checked C.

[1] https://github.com/microsoft/checkedc

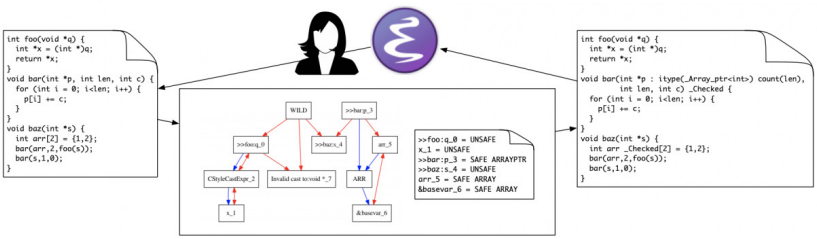

Figure 1: Workflow of using **3C** to refactor a program from C to Checked C.

Michael W. Hicks is a Professor in the Computer Science Department at the University of Maryland, where he is a Professor in the Computer Science Department at the University of Maryland, where he has been since 2002, and the CTO of Correct Computation, Inc. (CCI), a role he has held since late 2018.

He is a leading member of the software security community, having carried out diverse and award-winning research published in top venues covering computer security, programming languages, systems, and software engineering. He was the first Director of the University of Maryland's Cybersecurity Center (MC2), and was the elected Chair of ACM SIGPLAN, the Special Interest Group on Programming Languages; he now edits its blog, PL Perspectives, at https://blog.sigplan.org. Over his career, he has published more than 125 peer-reviewed scientific articles, and his research has twice won the NSA's Best Scientific Cybersecurity Paper competition; he is the only two-time winner. A key recent focus at UMD and CCI has been to explore how to make legacy software written in low-level languages, like C, more secure. He has been collaborating on the Checked C project, carrying on a nearly 20-year arc of work that started with the Cyclone safe programming language, which itself was a strong influence on Rust, a next-generation language. He has also currently looking at synergies between cryptography and programming languages; techniques for better random (fuzz) testing and probabilistic reasoning; and high-assurance tools and languages for quantum computing.

ADAPTING TO DEMAND: SEL4 PROOFS AND PROOF ENGINEERING PRACTICE

Matthew Brecknell
Data61, CSIRO, Australia

In this talk, we'll discuss some of our experiences scaling the formal verification of the seL4 microkernel to meet customer demand for increasingly complex requirements.

The formal verification of the seL4 microkernel [1] broke new ground, showing that comprehensive verification of a complex software system was possible. That initial effort required solutions to many proof engineering problems, for example, how to decompose refinement proofs so that the work could be distributed across a team [2].

Since the successful deployment of seL4 in the DARPA HACMS program, and the 2014 open-source release, there has been a growing interest in using seL4 in applications across a range of CPU architectures and platform configurations. At the same time, the kernel has gained significant new features, including symmetric multiprocessing (SMP), and support for real-time mixed-criticality systems (MCS). Verification of new ports and features is work in progress.

This growth has presented our proof engineering team with new grand challenges. We are no longer building new proofs about a relatively stable system. Rather, we're now adapting a large collection of existing proofs to rapid and sometimes radical changes to that system. Multi-processing requires changing the execution model underlying specifications and proofs [3]. Mixed-criticality features add complex interactions to interprocess communication (IPC) and scheduling [4], the two primary functions of a microkernel, while also demanding proofs of more interesting properties. Supporting many CPU architectures and platform configurations requires better abstraction of platform-specific aspects, especially virtual memory structures.

In this talk, we'll look more closely at these challenges, from both technical and organisational perspectives. We'll discuss some of the approaches we've taken so far, as well as ideas we have to improve on these in the future. While much of this work is currently domain-specific, we think that we will be able to extract general principles that will be useful to other verification projects.

References:

[1] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch and Simon Winwood. seL4: Formal verification of an OS kernel. ACM Symposium on Operating Systems Principles, pp. 207–220, Big Sky, MT, USA, October, 2009.

[2] David Cock, Gerwin Klein and Thomas Sewell. Secure microkernels, state monads and scalable refinement. Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics, pp. 167–182, Montreal, Canada, August, 2008.

[3] Sidney Amani, June Andronick, Maksym Bortin, Corey Lewis, Christine Rizkallah and Joey Tuong. COMPLX: A verification framework for concurrent imperative programs. International Conference on Certified Programs and Proofs, pp. 138–150, Paris, France, January, 2017.

[4] Anna Lyons, Kent Mcleod, Hesham Almatary and Gernot Heiser. Scheduling-context capabilities: A principled, light-weight OS mechanism for managing time. EuroSys Conference, Porto, Portugal, April, 2018.

Matthew Brecknell works with the Trustworthy Systems group at CSIRO's Data61, developing proofs and verification tools for the seL4 microkernel and its ecosystem. He has also consulted with other organisations doing verification in this ecosystem, and is a member of the Technical Steering Committee of the seL4 Foundation. He has contributed to the verification of the x86-64 port of seL4, and new features for mixed-criticality real-time systems. He is currently working on binary verification for the RISC-V port of seL4.

KEYNOTE PRESENTATION
DISTRIBUTED AND TRUSTWORTHY AUTOMATED REASONING

Marijn Heule
Carnegie Mellon University

Distributed automated reasoning has become increasingly powerful and popular. This enabled solving very hard problems ranging from determining the correctness of complex systems to answering long-standing open questions in mathematics. The tools are based on a portfolio of solvers that share information or divide-and-conquer. The portfolio approach is effective on large formulas from industry, while divide-and-conquer shines on hard-combinatorial problems. Recently distributed solvers competed for the first time, with each tool running on 1600 cores in the Amazon cloud.

The talk presents an overview of progress in distributed automated reasoning and covers some successes, including the solutions---with proofs---of the Boolean Pythagorean Triples problem and Keller's conjecture. The proofs are gigantic, but they have been validated using a formally-verified proof checker. These results underscore the effectiveness of distributed automated reasoning to solve hard challenges arising in mathematics and elsewhere, while having stronger guarantees of correctness than pen-and-paper proofs.

Marijn Huele is an Associate Professor at Carnegie Mellon University and received his PhD at Delft University of Technology (2008). His contributions to automated reasoning have enabled him and others to solve hard problems in formal verification and mathematics. He has developed award-winning SAT solvers and his preprocessing and proof producing techniques are used in many state-of-the-art solvers.

GEOMETRIC PATH ENUMERATION METHODS FOR VERIFYING ReLU NEURAL NETWORKS

Stanley Bak
Stony Brook University / Safe Sky Analytics

Neural Networks are universal function approximation tools that learn from examples, and increasingly being used in perception and control applications. For safety and mission-critical needs, one needs assurance that a neural network will produce outputs that satisfy formal properties. Several recent methods have been proposed in this direction, including one class of geometric methods that propages sets of states through a neural network.

In this talk, we review neural network execution and discuss methods targeting feedforward neural networks with ReLU activation functions and verification input/output properties. In these methods, it is critical to do range estimation efficiently, in order to determine if sets need to be split or not. These are essentially doing path enumeration for a neural network, where branches occur when the input to a ReLU can be either positive or negative. We explore a series of optimizations to this basic approach, and provide a comparison with several state-of-the-art tools from academia on a version of the ACAS Xu autonomous collision avoidance system that has been compressed using a series of neural networks

Stanley Bak is an assistant professor in the Department of Computer Science at Stony Brook University investigating the verification of autonomy, cyber-physical systems, and neural networks. He strives to develop practical formal methods that are both scalable and useful, which demands developing new theory, programming efficient tools and building experimental systems.

Bak received a Bachelor's degree in Computer Science from Rensselaer Polytechnic Institute (RPI) in 2007 (summa cum laude), and a Master's degree in Computer Science from the University of Illinois at Urbana-Champaign (UIUC) in 2009. He completed his PhD from the Department of Computer Science at UIUC in 2013. He received the Founders Award of Excellence for his undergraduate research at RPI in 2004, the Debra and Ira Cohen Graduate Fellowship from UIUC twice, in 2008 and 2009, and was awarded the Science, Mathematics and Research for Transformation (SMART) Scholarship from 2009 to 2013. From 2013 to 2018, Stanley was a Research Computer Scientist at the US Air Force Research Lab (AFRL), both in the Information Directorate in Rome, NY, and in the Aerospace Systems Directorate in Dayton, OH. He helped run Safe Sky Analytics, a research consulting company investigating verification and autonomous systems, and performed teaching at Georgetown University before joining Stony Brook University as an assistant professor in Fall 2020.

SCALING CONTINUOUS VERIFICATION

Joey Dodds
Galois

In 2017, we gave a talk at HCSS on two formal Cryptography correctness proofs that we performed on Amazon Web Services production code. Those two proofs covered around 200 lines of security critical C. Since then, we have pushed our tools forward, and our proofs now cover nearly 3000 lines of AWS C code, including post-quantum algorithms. All of these proofs are included in the continuous integration system, meaning we have scaled our formal methods over both code size and real-world updating of the code that we've proved. This talk will discuss the techniques and technology needed for this 15x increase in verified code.

One effect has been a huge increase in code change and proof maintenance cost. One proof in particular, our TLS handshake proof, sees particularly high development. This proof has two backup specifications that we never expect to change, so if the proofs break they can often be updated by developers, using the safety nets of the backup specifications to be sure that the desired properties still hold.

To address larger and more complicated code, we've made a number of changes to our verification tools. One of the most valuable has been an analysis to automatically generate "specification skeletons" which will often prove that code is free from undefined behavior at the push of a button. That proof can then be extended manually with functional correctness. We have also made numerous improvements to our tools to address the increasing demands of verification of post-quantum cryptography code.

The computational and memory demands of post-quantum cryptography dwarf those of most of the current algorithms in use. Unsurprisingly, verifying these algorithms also becomes significantly more challenging. In this talk, we will discuss some of the challenges to verification that we saw in two Cryptographic algorithms that we have verified, BIKE and SIKE. We will explain how we were able to overcome those limitations to arrive at the complete proofs that are running continuously today.

Joey Dodds is a Principal Researcher at Galois, where he focuses on commercial assurance efforts. He co-leads Galois' efforts with Amazon's AWS, working on verification projects including s2n, post-quantum cryptography and others. He has also led Galois's implementation of the ElectionGuard protocol and cryptography. Recently he has been focusing on how Galois can continue to improve its commercial assurance offerings by making the tools and approaches we use both easier to use, and to understand.

END-TO-END VERIFICATION OF INITIAL AND TRANSITION PROPERTIES OF GR(1) DESIGNS GENERATED BY SALTY IN SPARK

Laura Humphrey, James Hamil, Joffrey Huguet
Air Force Research Laboratory

Manual design of control logic for reactive systems is a time-consuming and error-prone process. An alternative is to automatically generate controllers from high-level specifications using "correct-by-construction" synthesis approaches. Recently, there has been interest in synthesizing controllers from Generalized Reactivity(1) or GR(1) specifications, since computational complexity is relatively low. Several tools for synthesis from GR(1) specifications now exist and have been used to generate controllers for aircraft power distribution, software-defined networks, ground robots, and teams of unmanned aerial vehicles, to name a few.

Here, we discuss Salty, an open-source tool that aids synthesis from GR(1) specifications. Salty addresses three shortcomings of existing synthesis tools. First, it makes specifications easier to write and debug by supporting features such as richer input and output types, user-defined macros, common specification patterns, and specification optimization and sanity checking. Second, it produces software implementations of synthesized controllers in a variety of languages, rather than simply producing controller design descriptions. Third, though synthesis from GR(1) specifications is theoretically "correct-by-construction," errors in tool implementation can lead to errors in synthesized controllers, so we have extended Salty to produce controllers in SPARK. SPARK is both a language and associated set of verification tools that has the potential to enable "end-to-end'' verification of synthesized controllers with respect to their original GR(1) specifications. To date, SPARK is able to automatically verify that synthesized controllers of modest (though not trivial) size satisfy a subset of properties comprising their original GR(1) specifications, namely system initial and transition properties.

In this talk, we introduce GR(1), some of its applications, Salty, and SPARK. We then describe how we encode synthesized controllers in SPARK, along with the necessary contracts and assertions needed for SPARK to verify system initial and transition properties properties automatically,

i.e. without requiring additional annotations from the user. Since GR(1) specifications encode assumptions about the environment in which the controlled system operates, we discuss how we handle these assumptions in the implementation, since there are several reasonable choices. We also discuss ideas for increasing the size of synthesized controllers that can be verified by SPARK. And finally, we discuss possible approaches for verifying the full set of GR(1) properties, i.e. including liveness.

Laura Humphrey received her Ph.D. in Electrical & Computer Engineering from the Ohio State University in 2009, where she specialized in control systems. Immediately afterward, she joined the Air Force Research Laboratory, where she began studying and applying formal methods. Her interests include the development of formally verified algorithms and software implementations, especially in support of autonomy and human-automation systems involving unmanned air vehicles.

SCALABLE SOUND STATIC ANALYSIS OF REAL-WORLD C APPLICATIONS USING ABSTRACT INTERPRETATION

***Henny Sipma,** **Paul E. Black
**Kestrel Technology, **National Institute of Standards and Technology*

Much of critical software systems continues to be written in C, especially embedded and cyberphysical systems. Despite decades of research in formal verication and increases in the sophistication of bug-nding tools, C applications continue to be plagued by exploitable vulnerabilities. The problem: academic tools often don't scale; bug-nding tools are mostly based on heuristics and therefore inherently incapable of providing guarantees.

Kestrel Technology has been developing an approach that addresses both concerns: it scales and has a solid mathematical foundation based on abstract interpretation and therefore oers the possibility of full assurance. The approach has been implemented in the tool KT Advance, a sound static analysis tool for undened behavior of C programs, including memory safety violations, integer oveflow, etc, covering more than 50 CWEs.

The approach consists of two stages: (1) automatic generation of proof obligations for all constructs in the C language that may have undened behavior, and (2) semi-automatic discharge of proof obligations using invariants generated by abstract interpretation. The challenge is the level of automation that can be achieved in the latter stage. The reward is full assurance accompanied by complete evidence that can be inspected and audited, with additional benets of automatically generated program documentation on API constraints and guarantees.

Scalability is achieved by compositionality: invariant generation, the computationally most expensive step in the analysis, is performed at the function level and its results are percolated up using assumeguarantee reasoning. This allows massive parallelization of the invariant generation. Much of the other reasoning can be done in parallel as well. The tool has been applied to applications with hundreds of thousands of lines of code, completing the analysis in an hour on a 16-processor server. We will present a case study of the analysis of an open-source drone navigation application of more than 100,000 lines of code, for which we were able to achieve 93% automatic discharge of all of the proof obligations.

KT Advance has been successfully exercised at NIST where it has been valuable in the SATE VI Ockham evaluation and clarication of the Juliet C test suites. The fact that KT Advance starts from a fundamental model of (valid) memory accesses and type conversions was especially helpful: it means that we can be sure that all possible kinds of, say buer overflows are examined, and that, if KT Advance does not report a violation, we can see exactly what it did consider. At NIST KT Advance was extended to produce OASIS Static Analysis Results Interchange Format (SARIF) v2.1 standard output.

This allows analysis results to be integrated in a larger tool chain or to be compared with results from other tools. We give our views on SARIF.

KT Advance is open-source (with MIT License) and available on GitHub and as such provides a low-cost path for government agencies and industry to gain the benet of the application of sound analysis and formal methods for real-world applications.

Henny Sipma has been working in formal methods and static analysis for more than 25 years, both in academia, at Stanford, where she did her PhD with Zohar Manna, and in industry, at Kestrel Technology, where she was one of the main developers of the CodeHawk tool suite.

ONE-CLICK AUTOMATED REASONING IN AMAZON WEB SERVICES

Andrew Gacek
Amazon Web Services

Amazon Web Services (AWS) recently launched IAM Access Analyzer, an automated reasoning service for auditing permissions to cloud resources. In this talk, we share the journey of bringing this formal methods technology to market. This includes wrestling with notions of correctness, getting users to specify correctness, and rethinking what correctness means. The result is a service based on formal methods yet accessible to everyday users.

Access Analyzer is built on top of Zelkova, an SMT-based analysis engine for AWS access control policies. The breakthrough in Access Analyzer is using predicate abstraction to provide a sound, precise, and compact summary of an access control policy. This summary enables compositional reasoning properties that are not possible of policies written in the underlying policy language.

Andrew Gacek is a Senior Applied Scientist at Amazon Web Services (AWS). Over the last two years, Andrew has developed techniques to apply automated reasoning to the identity and access control domain. Prior to AWS, Andrew spent seven years as an industrial logician in the Trusted Systems group at Rockwell Collins. There, Andrew applied automated reasoning to the development and verification of safety critical flight control systems. Andrew holds a PhD in Computer Science from the University of Minnesota.

CODE-LEVEL MODEL CHECKING IN THE SOFTWARE DEVELOPMENT WORKFLOW

Daniel Schwartz-Narbonne, Nathan Chong, Byron Cook, Konstantinos Kallas, Kareem Khazem, Felipe R. Monteiro, Serdar Tasiran, Michael Tautschnig, Mark R. Tuttle

Amazon Web Services

This experience report describes a style of applying symbolic model checking developed over the course of four years at Amazon Web Services (AWS). Lessons learned are drawn from proving properties of numerous C-based systems, e.g., custom hypervisors, encryption code, boot loaders, and the FreeRTOS operating system. Using our methodology, we find that we can prove the correctness of industrial low-level C-based systems with reasonable effort and predictability. Furthermore, AWS developers are increasingly writing their own formal specifications. All proofs discussed in this paper are publicly available on GitHub.

This is a paper about making code-level proof via model checking a routine part of the software development workflow in a large industrial organization. Formal verification of source code can have a significant positive impact on the quality of industrial code. In particular, formally verified specifications of code provide precise, machine-checked documentation for developers and consumers of a code base. They improve code quality by ensuring that the program's implementation reflects the developer's intent. Unlike testing, which can only validate code against a set of concrete inputs, formal proof can assure that the code is both secure and correct for all possible inputs.

The key obstacle to rapid proof development is that proofs tend to be written by a separate specialized team and not the software developers themselves. The developer writing a piece of code typically has an internal mental model of their code that explains why, and under what conditions, it is correct. However, this model typically remains known only to the developer – at best, it may be partially captured through informal code comments and design documents. As a result, the proof team must spend significant effort to reconstruct the formal specification of the code they are verifying. This slows the process of developing proofs.

Over the course of four years developing code-level proofs in Amazon Web Services (AWS), we have developed a proof methodology that is resulting proofs with reasonable effort,

and projects whose time-lengths can be reasonably predicted. For example, using these techniques, one full-time verification engineer, plus two interns, were able to specify and verify 171 entry points over key 9 modules in the AWS C Common library over a period of 24 weeks. Our prediction was that the effort would take 25 weeks. All specifications, proofs, and related artifacts (such as continuous integration reports), described in this paper have been integrated into the main AWS C Common repository on GitHub, and are publicly available at https://github.com/awslabs/aws-c-common.

Daniel Schwartz-Narbonne is a Senior Applied Scientist in the AWS Automated Reasoning Group. Prior to joining Amazon, he earned a PhD at Princeton, where he developed a software framework to debug parallel programs. As a postdoc at New York University, he designed a tool that automatically isolates and explains the cause of crashes in C programs. At Amazon, he has been focusing on integrating formal reasoning into the industrial workflow, enabling the continuous verification of key AWS software. When he's not working, you might find Daniel in the kitchen making dinner for his family, in a tent camping, or volunteering as an EMT with a local ambulance squad.

CONFERENCE POSTERS

A virtual poster session will be held from 2:00 p.m. to 3:30 p.m. on Tuesday, September 15.

bold name denotes presenter

CONFERENCE POSTERS

A New Kind of Program Logic

Lindsay Errington & Matt Sottile
Noddle

Affix: Micro-executing Binaries to Produce Static Analysis Models

Anwar Mamat, Ian Sweet, Michael Hicks
Correct Computation, Inc

Architecture Modeling for Resource Margin Estimation

***Srini Srinivasan**, *Russell Kegley, *Mark Gerhardt, *Rich Hilliard, **Clifford Granger, **Jonathan Preston, †Steven Drager, †Matthew Anderson , ††Richard Rosa, ††Alan Charsagua
**nHansa, **Lockheed Martin, † Air Force Research Labs, †† Naval Air Systems Command*

Memory Bugs Classes in the NIST Bugs Framework (BF)

Irena Bojanova & Carlos Galhardo
NIST

VisionGuard: Runtime Detection of Adversarial Inputs to Perception Systems

Yiannis Kantaros, Taylor Carpenter, Sangdon Park, Radoslav Ivanov, Sooyong Jang, Insup Lee, James Weimer
University of Pennsylvannia

A NEW KIND OF PROGRAM LOGIC

Lindsay Errington & Matt Sottile
Noddle

Noddle is building a code synthesis technology called ``pyx''. Pyx is a generic tool that can be specialized to different domains. For a given domain, pyx enables developers to write high-level applications in Python and generate specialized code to run on a range of platforms. Code is generated using an using an infrastructure that combines source-to-source program transformation, inference and automated search to explore the design space of algorithms and representations.

Each instance of pyx is in populated with transformation rules. Rules are written in a conservative extension of Python and collectively encode the semantics of Python, design knowledge for the domain and strategies for mapping abstract code to a specific platform.

Synthesis, like verification, depends on being able to reason about both code and specifications. In the case of Python, that means imperative code. And for pyx, it means being able to discharge proof obligations that appear in transformation rules. We experimented with using conventional logics and Floyd-style annotations for writing specifications. Similarly, we looked at leveraging external tools for inference. In the end we concluded that it was unlikely that such approaches would be accessible to prospective users or lead to effective inference.

Instead we have developed a new kind of program logic. The logic has a surprisingly simple model-theory and proof theory and offers the prospect of being able to reason about code and produce evidence (ie. proofs) using abstractions and syntax that are familiar to developers. Work on a prototype prover for transforming and reasoning about Python is ongoing.

Lindsay Errinton is a co-founder at Noddle. He was formerly a member of the the research staff at Kestrel Institute, Galois and Sandia National Labs in Livermore. With a background in logic and semantics of programming languages, he builds tools for developing high-assurance and high-performance applications.

AFFIX: MICRO-EXECUTING BINARIES TO PRODUCE STATIC ANALYSIS MODELS

Anwar Mamat, Ian Sweet, Michael Hicks

Correct Computation, Inc

As a formal method, static program analysis is highly appealing: today's tools can reason about useful properties (e.g., freedom from memory safety violations) at a modest of precision even for large programs.

However, this statement is true only for source programs; (arbitrary) machine code is much harder to analyze precisely. To deal with machine (aka binary) code, we have been developing a tool called Affix (Figure 1). It generates source-code models of binary programs which can be statically analyzed along with the source code that uses the binaries, thereby improving the results of the analysis (fewer false positives and negatives).

In this talk, we will describe how Affix works, and the steps we have taken to make its algorithm scale up. As a baseline, Affix uses micro execution, a technique for interpreting incomplete binary programs. Affix runs the API functions of a binary (e.g., based on its header file), setting input parameters to unknown.

Whenever such an unknown is used, it is replaced by a type-correct but random value; e.g., if the unknown is a pointer, then dereferencing it generates memory (itself, unknown) that can be accessed. Affix performs a special kind of taint propagation on unknown values deriving from/to input/output parameters/returns, and from API functions of interest to the analysis, such as malloc, free, getenv, exec, etc. At the conclusion of micro execution, Affix analyzes the state of memory to derive flows of tainted values; from them, it generates a C-code model that approximates the observed behavior. This process works well; we have generated useful models for checking several interesting properties using Facebook's infer analyzer. We will demo Affix's process during the talk and show how the basic idea scales up.

Anwar Mamat is a senior lecturer at the Computer Science Department of the University of Maryland, and a part time software engineer at the Correct Computation. He is interested in cyber-physical systems and programming languages.

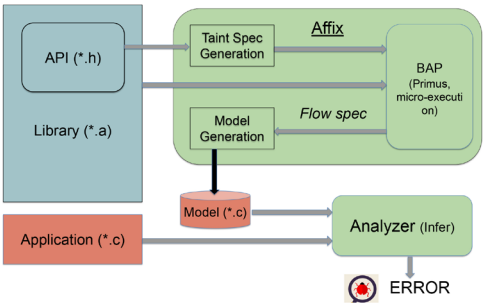

Figure 1: Affix: Its architecture and process for generating source-code models of binary code.

ARCHITECTURE MODELING FOR RESOURCE MARGIN ESTIMATION

***Srini Srinivasan,** *Russell Kegley, *Mark Gerhardt, *Rich Hilliard, **Clifford Granger, **Jonathan Preston, †Steven Drager, †Matthew Anderson, ††Richard Rosa, †Alan Charsagua

**nHansa, **Lockheed Martin, †Air Force Research Labs, ††Naval Air Systems Command*

Multi-core platforms induce implicit resource sharing, especially in memory and IO handling. Since these mechanisms are not arbitrated based on application priorities, they affect performance models and their analysis. Here, we illustrate how an architecture model constructed for performance analysis can be adjusted for multi-core resource sharing and estimate available margins. The adjustments are based on the construction of a multi-dimensional region derived from anticipated resource consumption patterns and locating an application's operating point within that region.

Srini Srinivasan is currently the founder and CTO of nHansa Inc of San Jose, CA. nHansa is engaged in the R&D and commercialization of software tools for embedded real-time systems, targeted for the design and analysis of performance- and safety-critical applications, such as aerospace and autonomous systems. Srini has over 30 years of experience in the embedded real-time systems domain both as a practitioner and as a tool vendor. Early in his career as a practitioner, he played an instrumental role in the development of an automated safety system for a nuclear power plant, and subsequent safety certification for operation. He was then a founder and the CEO of TimeSys Corporation, a vendor of schedulability analysis tools, Real-time Java and Real-time Linux products.

MEMORY BUGS CLASSES IN THE NIST BUGS FRAMEWORK (BF)

***Irena Bojanova, **Carlos Galhardo**

**National Institute of Standards and Technology (NIST), ** Brazilian National Institute of Metrology, Quality and Technology (INMETRO)*

The NIST Bugs Framework (BF) is an orthogonal classification of software bugs. A precise BF description of a software vulnerability would reveal the key steps for remediation of this vulnerability. Currently, BF covers Injection (INJ), Control of Interaction Frequency (CIF), Cryptography Bugs (ENC, VRF, KMN), Randomness Bugs (PRN, TRN), and Memory Bugs (MAD, MAL, MUS, MDL). In this presentation, we discuss our newly developed Memory Bugs cluster, consisting of the following BF classes: Memory Address Bugs (MAD), Memory Allocation Bugs (MAL), Memory Use Bugs (MUS), and Memory Deallocation Bugs (MDL). We present the BF Memory Bugs model, the causes-attributes-consequences graphs, and illustrative BF descriptions of specific CVEs.

Irena Bojanova is a computer scientist at the National Institute of Standards and Technology (NIST) and the PI of the Bugs Framework (BF) project. She earned her Ph.D. in Mathematics/ Computer Science from the Bulgarian Academy of Sciences. Irena is a Senior member of IEEE CS and serves as the Editor in Chief (EIC) of the IEEE IT Professional magazine.

Carlos E. C. Galhardo is a researcher at the Brazilian National Institute of Metrology, Quality and Technology, INMETRO. He is working at NIST as a guest researcher with the SAMATE--BF team. He earned his Ph.D in Physics from Universidade Federal Fluminense. His research interests include data analysis, physics of information and software security in embedded systems (measurement instruments).

VISIONGUARD: RUNTIME DETECTION OF ADVERSARIAL INPUTS TO PERCEPTION SYSTEMS

Yiannis Kantaros, Taylor Carpenter, Sangdon Park, Radoslav Ivanov, Sooyong Jang, Insup Lee, James Weimer
University of Pennsylvannia

Deep neural networks (DNNs) have been deployed in multiple safety-critical systems, such as medical imaging, autonomous cars, and surveillance systems. At the same time, DNNs have been shown to be vulnerable to adversarial examples [1], i.e., inputs which have deliberately been modified to cause either misclassification or desired incorrect prediction that would benefit an attacker.

In this work, to establish secure and high-confidence DNNbased perception systems, we propose VisionGuard, an attackand dataset-agnostic detection framework for defense against adversarial input images. VisionGuard relies on two key observations. First, adversaries take a real image and generate an attacked image by exploiting the large feature space over which they can look for adversarial inputs. We have validated this observation in our experiments: the larger the input space (i.e., image dimensions), the easier to fool the target classifier.

Second, we observe that the computer vision community has 50+ years experience in designing compression algorithms for real images – not attacked images. Thus, we expect real images to be better reconstructed by lossy compression than attacked images. Utilizing these two observations, VisionGuard effectively shrinks the feature space available to adversaries by processing both the original (possibly attacked) image and a 'refined' version generated through lossy compression (e.g., JPEG) with high compression quality. To determine if an image is adversarial, VisionGuard checks if the softmax output of the target classifier on a given input image changes significantly after feeding it a 'refined' version of that image. In VisionGuard, We measure the similarity of the corresponding softmax outputs using the Kullback - Leibler (K-L) divergence metric. If this metric is above a threshold, the image is classified as adversarial; otherwise, it is classified as clean. Conveniently, VisionGuard does not modify the specific classifier and does not rely on building separate classifiers; as such, the proposed approach can be used in coordination with existing defenses against adversarial examples such as adversarial/robust training [2] and image purification [3].

Similar defenses that rely on image transformations have also been proposed [4], [5]. For instance, MagNet [4], instead of compression algorithms, employs auto-encoders to generate new images that are reconstructed from the original ones.

Nevertheless, MagNet is dataset-specific as it requires training a new autoencoder for each dataset, a task that is particularly challenging and computationally demanding especially for large image domains. Image transformations have also been employed in [5] to detect adversarial inputs but in a completely different way than the proposed one. In particular, [5] relies The authors are with the School of Engineering and Applied Sciences, University of Pennsylvania, Philadelphia, PA 19103, USA, fkantaros,carptj,sangdonp,rivanov,lee,weimerjg@seas.upenn.edu. on building a DNN-based detector that takes as input K N features, where K is the number of applied transformations (e.g., rotation and translation) and N is the number of logits/ classes. Note that [5] is significantly more computationally expensive than VisionGuard both at run- and design- time.

Particularly, at design time [5] requires to train a DNN for the dataset of interest and at runtime, it requires application of K image transformations and the last hidden layer output for each transformed image. Alternative detectors have been proposed in [6], [7], that require extracting and storing the last hidden layer output for all training images which is a time-consuming and dataset-specific process and may not be possible on all platforms (e.g., lightweight IoT cameras) due to excessive memory requirements. These embeddings are used at runtime to check if an image is adversarial. Common in the above detectors is that they are dataset-specific. Therefore, it is unclear how these methods perform when they are deployed in real-world environments for which datasets do not exist. Also, the above detectors have only been evaluated on small-scale datasets such as MNIST and CIFAR10.

We evaluate VisionGuard on the MNIST, CIFAR10, and ImageNet datasets and we show that, unlike relevant works, it is very computationally light in terms of runtime and memory requirements, even when it is applied to large-scale datasets, such as ImageNet; therefore, it can be employed in real-time applications that may also involve large-scale image spaces.

For instance, the training process of the detectors in [6], [7] for the ImageNet dataset required 38 hours approximately and their performance on ImageNet is comparable to the performance of a random detector.

To the best of our knowledge, VisionGuard is the first attackagnostic and dataset-agnostic detection technique for defense against adversarial examples. Also, VisionGuard is the first detector that scales to large image domains (e.g., ImageNet dateset) while attaining high detection performance under a wide range of attacks – e.g., the area under the Receiver Operating Characteristics curve (AUC) is always greater than 90%.

References:

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199, 2013.

[2] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[3] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "Shield: Fast, practical defense and vaccination for deep learning using jpeg compression," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2018, pp. 196–204.

[4] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 135–147.

[5] S. Tian, G. Yang, and Y. Cai, "Detecting adversarial examples through image transformation," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[6] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," arXiv preprint arXiv:1703.00410, 2017.

[7] T. Pang, C. Du, Y. Dong, and J. Zhu, "Towards robust detection of adversarial examples," in Advances in Neural Information Processing Systems, 2018, pp. 4584–4594.

Yiannis Kantaros received the Diploma in Electrical and Computer Engineering in 2012 from the University of Patras, Patras, Greece. He also received the M.Sc. and the Ph.D. degrees in Mechanical Engineering from Duke University, Durham, NC, in 2017 and 2018, respectively. He is currently a postdoctoral research in the Department of Computer and Information Science at the University of Pennsylvania. His research focuses on distributed control, machine learning and formal methods with applications to distributed robotics. He received the Best Student Paper Award at the 2nd IEEE Global Conference on Signal and Information Processing in 2014 and the 2017-18 Outstanding Dissertation Research Award from the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC.