2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion

Rethinking Verification: Accuracy, Efficiency and Scalability through Human-Machine Collaboration*

Suresh Kothari
Iowa State University
Ames, Iowa
kothari@iastate.edu

Ahmed Tamrawi
Iowa State University
Ames, Iowa
atamrawi@iastate.edu

Jon Mathews
EnSoft Corp.
Ames, Iowa
jmathews@ensoftcorp.com

ABSTRACT

With growing dependence on software in embedded and cyber-physical systems where vulnerabilities and malware can lead to disasters, efficient and accurate verification has become a crucial need for safety and cybersecurity. Formal verification of large software has remained an elusive target, riddled with problems of low accuracy and high computational complexity [9, 11, 16, 18]. The need for automating verification is undoubted, however human is indispensable to accurate real-world software verification. The automation should actually enable and simplify human cross-checking, which is especially important when the stakes are high. This technical briefing discusses the challenges of creating a powerful fusion of automation and human intelligence to solve software verification problems where complete automation has remained intractable. We will contrast with existing software verification approaches and reflect on their strengths and limitations as a human-machine collaboration framework and outline key software engineering research and practice challenges to be addressed in the future.

1. INTRODUCTION TO THE TOPIC

The challenges of verifying our software infrastructure are daunting, in part because of the complexity of the software, but also due to the sheer volume of it. The Linux kernel alone, which provides the basis for so many devices (web servers, routers, smart phones, desktops), is over 12 MLOC. How can we verify this mountain of code?

Formal verification has been the holy grail of software engineering research [13]. Automated software verification methods have led to advances in data and control flow analyses, and applications of techniques such as Binary Decision Diagrams (BDDs) to analyze large software [15]. However, there are two fundamental limitations: (A) a completely automated and accurate analysis encounters NP hard problems, and (B) formal verification methods work mostly as automated black boxes with little support for cross-checking [2, 3, 5, 17, 19].

This technical briefing is about targeting automation to amplify human intelligence to scale it to large software. It projects the Intelligence Amplification (IA) vision propounded by Frederick Brooks [10]: *"If indeed our objective is to build computer systems that solve very challenging problems, my thesis is that IA > AI, that is, that intelligence amplifying systems can, at any given level of available systems technology, beat AI systems. That is, a machine and a mind can beat a mind-imitating machine working by itself."* The briefing will highlight the new frontier of software verification to ensure safety and security of critical software systems. It will bring out the key software verification research and practice challenges to be addressed in the future.

2. RELEVANCE TO SOFTWARE ENGINEERING COMMUNITY

The Intelligence Amplification vision is especially relevant in the context of software verification for cybersecurity and safety, where the the verification problems are extremely challenging for either the human or the machine to solve, and failures have catastrophic consequences. With this in mind, the recent DARPA STAC Program [6] for side channel and algorithmic complexity vulnerabilities calls for automated tools for human-in-the-loop software verification. With this briefing, researchers will benefit from an overview of the state-of-the-art research in this area while practitioners will benefit from an overview of the existing set of tools and techniques and their maturity for handling these challenges. Both audiences will benefit from discussion of open problems and the challenges as well as the opportunities they present for next generation software verification approaches.

3. TECHNICAL BRIEFING TOPIC

The briefing will discuss key challenges for an integrated human-machine approach to software verification from the following perspectives: (a) Human-Machine Collaborative Verification, (b) Evidence and Concept Empowered Verification, and (c) Interactive and Programmable Verification.

Human-Machine Collaboration: The machine and human have different strengths and weaknesses to complement each other. Verifying software for safety and cybersecurity vulnerabilities can be like looking for the needle in the haystack, not knowing what the needle looks like. The vulnerabilities must be hypothesized before they can be verified. The hypothesis is best originated from a human leveraging

*This material is based on research sponsored by DARPA under agreement numbers FA8750-15-2-0080 and FA8750-12-2-0126. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

ICSE '16 Companion, May 14 - 22, 2016, Austin, TX,USA

ACM ISBN 978-1-4503-4205-6/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2889160.2891046

the domain knowledge, but the human requires help from the machine to comb through large software to develop a hypothesis and then prove or reject it. The hypothesis would be in the form of a well-defined property (e.g., specification of a sensitive information leak) which can either be proved automatically or by a combination of human reasoning and automated analysis [14].

Evidence and Concept-Empowered Verification: The verification should be automated wherever possible, and complemented by human reasoning wherever needed. This calls for a mathematically rigorous notion of evidence that the machine generates to help the human to: (a) cross-check an automatically verified instance, and to (b) complete the verification where automation falls short. The evidence should reflect the hardness of each instance. For example, a recent version of Linux has more than 22 thousand instances of `mutex` and `spin` LOCK instances. The hardness of verifying that a LOCK is followed by an UNLOCK varies significantly among the multitude of LOCK instances because of factors such as the number of execution paths, the number of interacting functions, or whether the interaction happens directly or indirectly (e.g., a call using a function pointer). To avoid state explosion, a human is needed to discover domain-specific concepts to simplify proofs. For example, the state space for the computer-aided proof of the *four color problem* explodes without the simplifications due to the two key concepts of: *Discharging* [7] and *Reducibility* [8].

Interactive and Programmable Verification: Human-machine verification can benefit greatly from an *experiment-discover* paradigm. There are many opportunities to apply this paradigm; for example, the discovery of domain-specific simplifications or discovering the target properties for safety and cybersecurity. In this paradigm, a human interacts with large software by conceptualizing graph models of the inner workings of the software and uses those models to reason about the software. The software can be viewed as a large graph database of program artifacts and relationships between artifacts. The domain-specific concepts as well as verification-critical evidence can be formulated with the help of graph models. Graph database queries can be designed to enable interactive and query-embedded programming.

Future Challenges and Opportunities: After examining the existing verification approaches with respect to the above aspects, the briefing will conclude with a discussion of open challenges to be addressed by software verification research and practice. Examples of future challenges include: a mathematically rigorous notion of automatically generated verification-critical evidence, domain-specific simplifications that can avoid the inevitable state explosion in a completely automated generic verification, and a powerful query language for interactive and programmable verification.

Evidence and concept-empowered verification presents intriguing opportunities to improve accuracy and minimize human efforts for verification. For instance, consider problematic scenarios where automation cannot produce unambiguous evidence and would otherwise generate inaccurate conclusions, the automation could instead present such scenarios for the human to handle. The human can then gather additional information through interactive queries to resolve the ambiguity. The net result can be better accuracy and less human effort, because the interactive process may still be more efficient than triaging inaccurate and ambiguous reports from less sophisticated automation. Specifically, we will introduce the concepts: *accuracy boundary-aware* and *composable* analyzers and elaborate with concrete examples how these concepts could be central to accurate, efficient and scalable verification.

4. AUTHOR BIOGRAPHIES

Suresh Kothari is the Richardson Professor in the Electrical and Computer Engineering (ECE) department at Iowa State University. He has served as a PI on multi-million dollar DARPA APAC [1] and STAC [6] projects. He founded EnSoft [4] in 2002. EnSoft provides software safety and security products and services worldwide to avionics, automobile, and other Fortune 500 companies. Kothari has served as a Distinguished ACM Lecturer. He led the effort to establish a highly successful software engineering undergraduate degree program at Iowa State University. He was awarded in 2012 the Iowa State Board of Regents Professor Award for excellence in research, teaching, and service.

Ahmed Tamrawi is a Predoctoral Fellow at Iowa State University. Tamrawi has worked on the DARPA APAC and the DARPA STAC projects. He did his Ph.D. research on verification of safety properties of the Linux kernel.

Jon Mathews is the Chief Architect of the Atlas [12], the platform used by the Iowa State team to build security toolboxes for the DARPA APAC and STAC programs. Jon Mathews has been one of the three key cybersecurity analysts on the DARPA projects.

5. REFERENCES

[1] Automated Program Analysis for Cybersecurity (APAC). https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-11-63/listing.html.

[2] Clang static analyzer. http://clang-analyzer.llvm.org/.

[3] Coverity static analysis. http://www.coverity.com.

[4] Ensoft corp. http://www.ensoftcorp.com.

[5] Linux driver verification tool. http://linuxtesting.org/ldv.

[6] Space/Time Analysis for Cybersecurity (STAC). https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-14-60/listing.html.

[7] K. Appel and W. Haken. Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, 1977.

[8] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. Part II: Reducibility. *Illinois Journal of Mathematics*, 1977.

[9] D. Beyer and A. K. Petrenko. Linux driver verification. In *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*. Springer, 2012.

[10] F. P. Brooks Jr. The computer scientist as toolsmith II. *Communications of the ACM*, 39(3):61–68, 1996.

[11] C. Canal and A. Idani. *Software Engineering and Formal Methods: SEFM 2014 Collocated Workshops*. Springer, 2015.

[12] T. Deering, S. Kothari, J. Sauceda, and J. Mathews. Atlas: a new way to explore software, build analysis tools. In *Companion Proceedings of the 36th ICSE*, 2014.

[13] B. Gates. Bill Gates Keynote: Microsoft Tech-Ed, 2008.

[14] M. I. Gordon, D. Kim, J. H. Perkins, L. Gilham, N. Nguyen, and M. C. Rinard. Information flow analysis of android applications in droidsafe. In *NDSS*, 2015.

[15] O. Lhoták. *Program analysis using binary decision diagrams*. PhD thesis, McGill University, 2006.

[16] P. Stratis. Formal verification in large-scaled software: Worth to ponder? https://blog.inf.ed.ac.uk/sapm/2014/02/20/formal-verification-in-large-scaled-software-worth-to-ponder/.

[17] Y. Sui, D. Ye, and J. Xue. Detecting memory leaks statically with full-sparse value-flow analysis. *Software Engineering, IEEE Transactions on*, 40(2):107–122, 2014.

[18] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald. Formal methods: Practice and experience. *ACM Computing Surveys (CSUR)*, 41(4):19, 2009.

[19] Y. Xie and A. Aiken. Saturn: A scalable framework for error detection using boolean satisfiability. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 2007.