Whatever Happened to Formal Methods?

J.Voas and K. Schaffer (*Cybertrust* column article for IEEE *Computer* Magazine)

We asked 7 experts 1 simple question to find out what has occurred recently in terms of applying formal methods (FM) to security-centric, cyber problems:

Please summarize in a paragraph the state of the research and practitioner communities in formal method as you see it. Please include standards, conferences, tools, and anything else for our readers that they may not be aware of concerning recent noteworthy advancements and improvements in the FM community in recent years.

Our experts were John McLean (Naval Research Labs), Paul Black (National Institute of Standards and Technology), Karl Levitt (University of California at Davis), Joseph Williams (CloudEconomist.Com), Connie Heitmeyer (Naval Research Labs), Eugene Spafford (Purdue University), and Joseph Kiniry (Galois, Inc.). Our experts answered with unique personal insights as follows:

Karl Levitt

Traditionally, formal methods meant the use of the Floyd-Hoare method to verify program code with respect to assertions. Regarding the verification with respect to security, the assertions have usually captured the property of access control: processes (or other entities operating on behalf of users) are granted access to "objects" only if security properties captured as invariants are satisfied. Several "toy" operating systems were successfully verified as exemplary exercises, notably Bevier's KIT system.

To address leakage through "storage channels", the concept of information flow was introduced. The information flow property expresses allowed flows among users and/or objects; information verification supersedes access control verification but is usually more difficult to achieve. Numerous research efforts produced flow analyzers for real programming languages and applied the method to what are arguably real parts of operating systems and applications.

Starting in late-1970s, security verification was used to reason about abstractions of code, namely specifications; this approach was at the time called design verification, and avoided the tedious and error-prone step of capturing the semantics of complex programming languages as required by the Floyd-Hoare method. Over the years, the method was applied successfully to abstractions of real systems, often exposing storage channels. Even if they are not subject to formal security analysis, it has been claimed that the exercise of writing specifications has value in its own right in exposing design flaws although not necessarily flaws in the ultimate programs. Popular specification languages are Z and VDM, among others

Tools have been developed to support both kinds of verification.

Recently, there has been considerable effort in what we can call lightweight formal methods, which still entail formal reasoning but fall short of the assurance possible for "verification". These methods have benefitted significantly from recent advances in model checking and SMT which are reasoning methods in a decidable domain; these methods are now used in most verification systems to fully automated a substantial part of the reasoning.

Among the successes in so-called lightweight formal methods applied to security are:

- *Code checking for security flaws: Typically, these tools analyze code for the (possible) presence of common security flaws, such as unitialized variables, race conditions, buffer overflows. In principle, any code property that gives rise to vulnerabilities in code can be handled by these tools. To date, the tools have been used to analyze automatically the implementation of large operating systems OS often discovering previously unknown vulnerabilities. Some of the "vulnerabilities" found are false positives which require manual analysis to resolve.*
- *Automatic generation of test cases:*
- *Checking the rules of large firewalls: A firewall for a large enterprise can contain over 10000 rules. Tools to "check" the rulesets, largely for conflicts or redundancies, have been developed and applied successfully.*
- *Tools to analyze security protocols for flaws: Such tools have been used to confirm previously known flaw in key-exchange protocols but also to identify previously unknown flaws in implementations of Kerberos. Other research attempts to use reduction to reason about security protocols.*
- *Capturing the formal semantics of "security": Primarily through ontologies, the semantics of many attacks can be expressed. Attack graphs are a very important example, and through analysis it can often be determined what attacks a given system, characterized using rules, is subject to. Also, this kind of analysis has been used to "explain" the output of Snort intrusion detection systems.*

Joseph Williams

My introduction to Formal Methods was the 1996 summary article by Clarke and Wing [1] *that appeared in ACM Computing Surveys that I was hoping would help me puzzle out some problems I had building internet search crawlers. By 2001, RW Butler (NASA) was already observing that the enormous complexity of real systems made it difficult to apply formal methods. Bertrand Meyer (ETH Zurich) argued that formal methods were inevitable, but he wasn't sure how long it would take. Last year Sharif Omar Salem (Limkokwing University) observed that researchers are still trying to realize the benefits of using formal methods. There are some great case studies showing the value of formal methods, but my sense is that we still haven't arrived at Meyer's inevitable destination. I count at least 83 conferences this year that will have some focus on formal methods, so the field does not lack from attention. However, in this era of agile development and fast-to-fail business modeling, the rigor of formal methods seems still to fit best in safety systems and the analytical fields (data mining, cybersecurity) where researchers benefit from a longer view of the problem and solution space.*

John McLean

Formal Methods continue to have a robust presence in the computer science research area. Besides dedicated journals such as Formal Methods in System Design, there are also both established dedicated workshops such as The International Workshop on Formal Methods in Industrial Critical Systems (FMICS) [2]*, which is having its 23rd event this year, and new ones focused on special subareas such as the 2nd International Workshop on Symbolic and Numerical Methods for Reachability Analysis* [3]*. There are also a large number of dedicated conferences (eg, the International Symposium on Formal Methods* [4]*, NASA Formal Methods Symposium* [5]*, integrated Formal Methods (iFM)* [6]*, ABZ* [7]*, The International Conference on Software Engineering and Formal Methods* [8]*, etc) and conferences and workshops where Formal Methods play a key role (eg, The European Joint*

Conferences on Theory and Practice of Software (ETAPS), The Computer Security Foundations Workshop, etc). All of these feature a mix of theoretical advances, case examples, and formal method supporting technology, such as CAV tools.

Paul Black

Formal methods are slowly becoming more accepted and more widely used. Two conferences that I especially enjoy and that are less known are NASA Formal Methods (NFM) Symposium [5] *and High Confidence Software and Systems (HCSS) Conference* [9]*. NFM is typically in June in various cities in the United States. HCSS is typically in May and has been in Maryland, USA.*

Joseph Kiniry

Formal methods researchers are pursuing two complementary paths. The bulk of the community continues to be focused on foundations (what I'd call "pure FM"), and another part of the community looks for opportunities to use FM in practice ("applied FM"). These are not disjoint sets and I find that the latter group is growing over time as we see an increasing need for FM in industry, particularly for matters relating to critical systems and security.

In my experience, the bulk of applied FM in the "real world" is rarely reported in a workshop, conference, or journal. Those projects are most often witnessed indirectly through presentations or software releases.

When I was a full-time academic, I often wondered why this was the case. Now that I'm full-time in industry, I better understand.

For the most part, at least at my organization, this is not due to a lack of desire to share about our accomplishments; it is often simply a matter of time—time to write a paper or presentation, time to attend a conference, time to connect with my community. All of that time spent talking about the past is, in some sad sense, better spent solving the next problem.

The other conflicting factor is the constraints under which we work. Whether it is mutual non-disclosures with our customers or pre-publication review by defense agencies, it is often the case that we are not permitted to talk about FM-related work...or at least we cannot without a significant amount of wrestling with bureaucracies and lawyers. Once again, our time is better spent elsewhere.

This framing is evolving. Our default position on most new opportunities is one of transparency and openness. For example, we now insist that we would like to have the option to open source technologies and publish peer-reviewed papers. We find that our customers are more amenable to this strategy when they understand early-on the benefit to our businesses and society. Our releases of Cryptol and SAW are two successes in this vein.

Connie Heitmeyer

The state of the art in formal methods has advanced significantly since the introduction more than 40 years ago of these methods to reason about security and other critical properties of systems. Today, scores of conferences and workshops, such as FM (the International Symposium on Formal Methods), NFM (the NASA Formal Methods Symposium), CAV (the International Conference on Computer-Aided Verification), and SPIN (the International SPIN Symposium on Model Checking of Software), and many journals, including Formal Methods in System Design and Formal Aspects of Computing, publish a wide range of new and promising research results on formal methods topics such as model checking, theorem proving, static analysis, and decision procedures; on tools supporting formal modeling and analysis; and on applications of formal methods and tools to find flaws in, and to verify critical properties of, systems and software. In practice, hardware vendors, such as Intel, have for years used model

checkers and theorem provers to detect errors in and to verify hardware designs. Moreover, increasingly, formal methods are being used to model and to analyze the design of safety-critical and security-critical software systems. Representation applications of formal methods include the use of formal modeling and interactive theorem proving as evidence in certifying security-critical software devices (see, e.g., [10], [11]*) and of model checking to detect flaws in the design of safety-critical medical devices (see, e.g.,* [12]*). Unfortunately, however, although many researchers have successfully applied formal methods in software practice, the use of formal methods by software developers is rare.*

Gene Spafford

Most of the community I work with are implementers and incident response personnel. I see effectively no use of formal methods. In many cases, even simple defensive programming is not used. Few, if any, of the programmers and designers know anything about formal methods other than "it's expensive and slow." State-of-the-art for some of them is considering using the SDL and some static code checkers!

Summary

We thank our experts for their candidness. And we invite you to read their answers to 7 other similar questions in the *Perspectives* section of this issue.

References

[1] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Surveys (CSUR)*, vol. 28, no. 4, pp. 626–643, 1996.

[2] (unknown author), "Formal Methods for Industrial Critical Systems," 2016. [Online]. Available: http://fmics.inria.fr/. [Accessed: 11-Mar-2016]

[3] "2nd International Workshop on Symbolic and Numerical Methods for Reachability Analysis," 2016. [Online]. Available: https://snr2016.pages.ist.ac.at/. [Accessed: 11-Mar-2016]

[4] "21st International Conference on Formal Methods," 2016. [Online]. Available: http://fm2016.cs.ucy.ac.cy/. [Accessed: 11-Mar-2016]

[5] "NASA Formal Methods Symposium," 2016. [Online]. Available: http://shemesh.larc.nasa.gov/NFM/. [Accessed: 11-Mar-2016]

[6] "iFM 2016," 2016. [Online]. Available: http://en.ru.is/ifm. [Accessed: 11-Mar-2016]

[7] "ABZ 2016," 2016. [Online]. Available: http://www.cdcc.faw.jku.at/ABZ2016/. [Accessed: 11-Mar-2016]

[8] "SEFM 2016," 2016. [Online]. Available: http://staf2016.conf.tuwien.ac.at/sefm/. [Accessed: 11-Mar-2016]

[9] "High Confidence Software and Systems Conference," 2016. [Online]. Available: http://cps-vo.org/group/hcss_conference. [Accessed: 11-Mar-2016]

[10] D. Greve, M. Wilding, and W. M. Vanfleet, “A separation kernel formal security policy,” in *Proc. Fourth International Workshop on the ACL2 Theorem Prover and Its Applications*, 2003.

[11] C. L. Heitmeyer, M. M. Archer, E. I. Leonard, and J. D. McLean, “Applying formal methods to a certifiably secure software system,” *Software Engineering, IEEE Transactions on*, vol. 34, no. 1, pp. 82–98, 2008.

[12] Z. Jiang, M. Pajic, R. Alur, and R. Mangharam, “Closed-loop verification of medical devices with model abstraction and refinement,” *International Journal on Software Tools for Technology Transfer*, vol. 16, no. 2, pp. 191–213, 2014.