



Insights on Formal Methods in Cybersecurity

Jeffrey Voas, IEEE Fellow

Kim Schaffer, National Institute of Standards and Technology

Seven experts weigh in on the current use and practice of formal methods in cybersecurity.

We asked seven experts to summarize the state of the art and practice in formal methods (FM) as applied to cybersecurity. In particular, we asked them to identify noteworthy standards, conferences, tools, and anything else *Computer* readers might not be aware of that have significantly contributed to FM advances in recent years.

Our experts were Karl Levitt of the University of California, Davis; Joseph Williams of CloudEconomist.com; John McLean and Connie Heitmeyer of the US Naval Research Laboratory; Paul Black of the National Institute of Standards and Technology (NIST); Joseph Kiniry of Galois, Inc.; and Eugene Spafford of Purdue University. Their unique personal insights are summarized below, with more detailed comments to appear in an August cover feature.

KARL LEVITT

Traditionally, FM meant the use of the Floyd-Hoare method to verify program code with respect to assertions. In terms of 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” OSs were successfully verified as exemplary exercises, notably William Bevier’s Kit (kernel for

isolated tasks) system.¹

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 OSs and applications.

Starting in the late 1970s, *security verification* was used to reason about abstractions of code, namely specifications; at the time this approach was 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 specifications aren’t subject to formal security analysis, it has been claimed that the exercise of writing them has



value in its own right by exposing design flaws, although not necessarily flaws in the ultimate programs. Popular specification languages include Z and VDM [Vienna Development Method]-SL, among others.

Tools have been developed to support both information and security verification.

Recently, there has been considerable effort in what we can call *light-weight* formal methods, which still entail formal reasoning but fall short of the assurance possible for “verification.” These have benefitted significantly from recent advances in model checking and SMT (satisfiability modulo theories), which are reasoning methods in a decidable domain; such methods are now used in most verification systems to fully automate a substantial part of the reasoning.

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

- › Code-checking tools for security flaws. Typically, these tools analyze code for the (possible) presence of common security flaws, such as uninitialized variables, race conditions, and 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 automatically analyze the implementation of large OSs, often discovering previously unknown vulnerabilities. Some of the “vulnerabilities” found are false positives that require manual analysis to resolve.
- › Automatic generation of test cases.
- › Tools to check the rules of large firewalls. A firewall for a large enterprise can contain more than 10,000 rules. Tools to check

the rule sets, 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 flaws in key-exchange protocols but also to identify previously unknown flaws in Kerberos implementations. 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 FM was the 1996 summary article by Edmund Clarke and Jeannette Wing in *ACM Computing Surveys*,² which helped me puzzle out some problems I had building Internet search crawlers that couldn’t be effectively tested empirically on every host configuration before the crawlers were released.

In 2002, Ricky Butler (NASA), along with Rick Kuhn and Ramaswamy Chandramouli (NIST), observed that the enormous complexity of real systems made it difficult to apply FM.³ That same year, Bertrand Meyer (ETH Zurich), who had first mused about FM in 1985⁴ and continued to elaborate on it throughout his development of Eiffel, argued in a piece with Karine Arnout that FM was inevitable but that he wasn’t sure how long it would take.⁵

Some excellent case studies demonstrate the value of FM, but my sense is that we still haven’t arrived at Meyer’s “inevitable” destination. I counted at least 83 conferences being held this year that will have some focus on FM, so the field doesn’t lack attention. However, in this era of agile development and fast-to-fail business modeling, the rigor of FM still seems to fit best in safety-critical systems and in analytical fields such as data mining and cybersecurity, where researchers benefit from a longer view of the problem and solution space.

JOHN MCLEAN

FM continues to have a robust presence in the computer science research area.

Besides dedicated journals such as *Formal Methods in System Design*, there are numerous annual FM conferences including the International Symposium on Formal Methods (FM; <http://fm2016.cs.ucy.ac.cy>), the NASA Formal Methods Symposium (NFM; <http://shemesh.larc.nasa.gov/NFM>), the International Conference on Integrated Formal Methods (iFM; <http://en.ru.is/ifm>), the International ABZ Conference (ABZ; www.cdcc.faw.jku.at/ABZ2016), and the International Conference on Software Engineering and Formal Methods (SEFM; <http://staf2016.conf.tuwien.ac.at/sefm>).

There are also established workshops on subspecialty areas, such as the International Workshop on Formal Methods for Industrial Critical Systems (FMICS), which is holding its 21st event in 2016 (<http://fmics-avocs.isti.cnr.it>), as well as relatively new ones, such as the International Workshop on Symbolic and Numerical Methods for Reachability Analysis (SNR), which is meeting for the second time this year (<https://snr2016.pages.ist.ac.at>).

FM plays a key role in many other conferences and workshops, including the European Joint Conferences

on Theory and Practice of Software (ETAPS; www.etaps.org) and the Computer Security Foundations Symposium (CSF; www.ieee-security.org/CSFWweb).

All of these gatherings feature a mix of theoretical advances, use cases, and FM-supporting technology, such as computer-aided verification (CAV) tools.

PAUL E. BLACK

Formal methods are gradually becoming more accepted and widely used. For instance, standards bodies typically require formal, automated proofs of secure communication or key-exchange protocols before approval.

in Annapolis, Maryland. Neither conference charges for registration.

JOSEPH KINIRY

FM researchers are pursuing two complementary paths. The bulk of the community continues to focus on foundations (what I call “pure FM”), while the rest of the community looks for opportunities to use FM in practice (“applied FM”). These aren’t disjoint groups, and I find that the latter group is growing over time in response to the increasing need for FM in industry, particularly in matters relating to critical systems and cybersecurity.

In my experience, the bulk of applied FM in the “real world” is rarely

This framework is evolving. Our company’s default position on most new opportunities is openness and transparency. For example, we now insist on the option to use open source technologies and on publishing peer-reviewed papers. As evidenced by the success of two of our releases, Cryptol (www.cryptol.net) and SAW (Software Analysis Workbench; <https://galois.com/project/software-analysis-workbench>), our customers are more amenable to this strategy when they understand early on its benefit to both business and society in general.

CONNIE HEITMEYER

The state of the art in formal methods has advanced considerably since the introduction more than 40 years ago of these methods to reason about security and other critical system properties. Today, scores of conferences and workshops—such as FM, NFM, the International Conference on Computer-Aided Verification (CAV; <http://i-cav.org>), and the International SPIN Symposium on Model Checking of Software (SPIN; www.spin2016.info)—and many journals, including *Formal Methods in System Design* and *Formal Aspects of Computing*, publish a wide range of new and promising FM research results. Topics include model-checking, theorem-proving, static-analysis, and decision procedures; tools supporting formal modeling and analysis; and FM applications 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 verify hardware designs. Moreover, formal methods are increasingly being used to model and analyze the design of safety- and security-critical software systems. Representative applications include the use of formal modeling and interactive theorem proving as evidence in certifying security-critical software devices^{6,7} and of model checking to detect flaws in the design of medical devices.⁸

Formal methods are increasingly being used to model and analyze the design of safety- and security-critical software systems.

Compilers use formally verified algorithms for register allocation and code optimization. In solving big data and distributed computing problems, such as count-distinct, researchers continue to deliver proofs with a high probability of success.

Many current static analysis tools use heuristics and approximations to check millions of lines of code (LOC) for dozens of classes of weaknesses while producing low false-alarm rates. Some static analyzers—including SPARK Pro, Frama-C, and CodeHawk—use formal methods to produce always-correct results for hundreds of thousands of LOC.

Two annual FM conferences bring together an interesting mix of researchers and technologists using formal methods to solve real-world problems, from drone guidance to detecting race conditions: NFM, which is held in April or May in various cities around the US; and the High Confidence Software and Systems Conference (HCSS), which takes place in May

reported in a workshop, conference, or journal. Those projects are most often revealed 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 understand.


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

The other conflicting factor is the constraints under which we work. Whether due to mutual nondisclosures with our customers or prepublication review by defense agencies, often we aren’t permitted to discuss FM-related work—at least not without significant wrangling with bureaucracies and lawyers. Again, time is better spent elsewhere.

Unfortunately, although many researchers have successfully applied formal methods in software practice, their use by software developers is rare.

EUGENE SPAFFORD

Most of the community I work with are security software designers and IT incident response personnel. I see effectively no use of FM. In many cases, even simple defensive programming isn't used. Few, if any, of the programmers and designers know anything about FM other than that "it's expensive and slow." State of the art for some of them is considering using the Micro soft Security Development Lifecycle and some static code checkers!

We thank our experts for their candidness. Look for an expanded discussion of their responses in the August edition of *Computer*. 

REFERENCES

1. W.R. Bevier, "Kit: A Study in Operating System Verification," *IEEE Trans. Software Eng.*, vol. 15, no. 11, 1989, pp. 1382-1396.
2. E.M. Clarke and J.M. Wing, "Formal Methods: State of the Art and Future Directions," *ACM Computing Surveys*, vol. 28, no. 4, 1996, pp. 626-643.
3. D.R. Kuhn, R. Chandramouli, and R.W. Butler, "Cost Effective Use of Formal Methods in Verification and Validation," *Proc. Workshop Foundations for Modeling and Simulation Verification and Validation for the 21st Century (Foundations 02)*, 2002; http://csrc.nist.gov/groups/SNS/asft/documents/Foundations_2002.pdf.
4. B. Meyer, "On Formalism in Specifications," *IEEE Software*, vol. 3, no. 1, 1985, pp. 6-26.
5. K. Arnout and B. Meyer, "Finding Implicit Contracts in .NET Components," *Formal Methods for Components and Objects*, F.S. de Boer et al., eds., LNCS 2852, Springer, 2002, pp. 285-318.
6. D. Greve, M. Wilding, and W.M. Vanfleet, "A Separation Kernel Formal Security Policy," *Proc. 4th Int'l Workshop ACL2 Theorem Prover and Its Applications*, 2003; www.cs.utexas.edu/users/moore/acl2/workshop-2003/contrib/greve-wilding-vanfleet/security-policy.ps.
7. C.L. Heitmeyer et al., "Applying Formal Methods to a Certifiably Secure Software System," *IEEE Trans. Software Eng.*, vol. 34, no. 1, 2008, pp. 82-98.
8. Z. Jiang et al., "Closed-Loop Verification of Medical Devices with Model Abstraction and Refinement," *Int'l J. Software Tools for Technology Transfer*, vol. 16, no. 2, 2014, pp. 191-213.

JEFFREY VOAS, Cybertrust column editor, is an IEEE Fellow. Contact him at j.voas@ieee.org.

KIM SCHAFFER is an IT specialist in the Computer Security Division of the National Institute of Standards and Technology. Contact him at kim.schaffer@nist.gov.

ADVERTISER INFORMATION • MAY 2016

Advertising Personnel

Debbie Sims: Advertising Coordinator
Email: dsims@computer.org
Phone: +1 714 816 2138 | Fax: +1 714 821 4010

Chris Ruoff: Senior Sales Manager
Email: cruoff@computer.org
Phone: +1 714 816 2168 | Fax: +1 714 821 4010

Advertising Sales Representatives (display)

Central, Northwest, Southeast, Far East:
Eric Kincaid
Email: e.kincaid@computer.org
Phone: +1 214 673 3742
Fax: +1 888 886 8599

Northeast, Midwest, Europe, Middle East:

David Schissler
Email: d.schissler@computer.org
Phone: +1 508 394 4026
Fax: +1 508 394 1707

Southwest, California:

Mike Hughes
Email: mikehughes@computer.org
Phone: +1 805 529 6790

Advertising Sales Representative (Classifieds & Jobs Board)

Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 201 887 1703