NIST Special Publication 500-320

Report of the Workshop on Software Measures and Metrics to Reduce Security Vulnerabilities (SwMM-RSV)

Paul E. Black
Elizabeth Fong

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.500-320

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Special Publication 500-320

Report of the Workshop on Software Measures and Metrics to Reduce Security Vulnerabilities (SwMM-RSV)

Paul E. Black
Elizabeth Fong
Software and Systems Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.500-320

November 2016

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Under Secretary of Commerce for Standards and Technology and Director

National Institute of Standards and Technology Special Publication 500-320
Natl. Inst. Stand. Technol. Spec. Publ. 500-320, 84 pages (November 2016)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.500-320

Abstract

The National Institute of Standards and Technology (NIST) workshop on Software Measures and Metrics to Reduce Security Vulnerabilities (SwMM-RSV) was held on 12 July 2016. The goal of this workshop is to gather ideas on how the Federal Government can identify, improve, package, deliver, or boost the use of software measures and metrics to significantly reduce vulnerabilities.

This report contains observations and recommendations from the workshop participants. This report also includes position statements submitted to the workshop, presentations at the workshop, and related material. Ideas from the workshop will be included in the Dramatically Reducing Software Vulnerabilities report, requested of NIST by the White House Office of Science and Technology Policy in Spring 2016.

Keywords:

Measurement; metrics; software assurance; security vulnerabilities; reduce security vulnerabilities.

Disclaimer:

This report includes position statements and presentation slides by authors who submitted their material to the workshop. The views expressed by the authors therein do not necessarily reflect those of the sponsors of this workshop.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology (NIST), nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

Acknowledgement:

The authors thank Peggy Himes and Rose Linares of NIST for their valuable help with the workshop and formatting this manuscript. Thanks to Simson Garfinkel for significant contributions, which improved this report. Thanks also to the many workshop participants who contributed, discussed, refined, and wrote up ideas.

Contents

1. Overview

The 2016 Federal Cybersecurity Research and Development Strategic Plan [1] seeks to fundamentally alter the dynamics of security in the computer realm, reversing adversaries' asymmetrical advantages. The plan calls for "sustainably secure systems development and operation." To achieve this, the plan describes a mid-term (3-7 years) goal of "the design and implementation of software, firmware, and hardware that are highly resistant to malicious cyber activities…" and reduce the number of vulnerabilities in software by orders of magnitude.

The Software Quality Group at the U.S. National Institute of Standards and Technology (NIST) felt that measures of software can play an important role in such dramatic reductions. Industry requires evidence to indicate how vulnerable a piece of software is, know what techniques are most effective in developing software with far fewer vulnerabilities, determine the best places to deploy defensive measures, or take any of a number of other actions. This evidence comes from measuring, in the broadest sense, or assessing properties of software. If there were comprehensive metrics, it would be straight-forward to determine which software development technologies or methodologies lead to sustainably secure systems.

Accordingly, in Spring 2016 we decided to organize a workshop to gather ideas on how the Federal Government can identify, improve, package, deliver, or boost the use of software measures and metrics to significantly reduce vulnerabilities. We called for short position statements, one to three paragraphs long, to begin to gather ideas. We asked for position statements, rather than papers, to decrease the work required of submitters. In the call for statements, we suggested the following subjects:

- Existing measures of software that can make a difference in three to seven years,
- Means of validating software measures or confirming their efficacy (meta-measurements),
- Quantities (properties) in software that can be measured,
- Standards (in both *étalon* and *norme* senses) needed for software measurement,
- Cost vs. benefit of software measurements,
- Surmountable barriers to adoption of measures and metrics,
- Areas or conditions of applicability (or non-applicability) of measures,
- Software measurement procedures (esp. automated ones), or
- Sources of variability or uncertainty in software metrics or measures.

Twenty positions statements were submitted. A program committee evaluated the submissions for relevance to the workshop theme and potential interest. The committee invited workshop presentations based on 10 statements.

Ideas from this workshop and other efforts will be included in the report on Dramatically Reducing Software Vulnerabilities, requested of NIST by the White House Office of Science and Technology Policy in Spring 2016.

The workshop was open to all, subject to the rules of entry to the NIST campus, and there was no cost to attend.

1.1 Mechanics and Organization

The workshop was co-chaired by Paul E. Black and Elizabeth Fong, National Institute of Standards and Technology and by Thomas D. Hurt, Office of the Deputy Assistant Secretary of Defense for Systems Engineering - Joint Federated Assurance Center (JFAC) lead.

The program committee consisted of Paul E. Black, David Flater, Elizabeth Fong, D. Richard Kuhn, and W. Timothy Polk.

The web site is https://samate.nist.gov/SwMM-RSV2016.html. In late April and early May, co-chairs sent the call for statements to mailing lists and many individuals. Here is the timeline:

22 May: deadline to submit statements
8 June: invitations to present sent
27 June: deadline for non-citizens to register
5 July: deadline for US citizens to register
12 July: workshop
31 July: deadline for submission of revised statement or presentation

1.2 Agenda and Schedule

The workshop was held at the National Institute of Standards and Technology, Gaithersburg, Maryland, on 12 July 2016. Over 90 people from Federal Government, software assurance tool makers, service providers, and universities attended. The program consisted of nine presentations invited on the basis of position statements and one breakout session. For the breakout, attendees were randomly assigned to one of six breakout groups. All the groups had the same charge: come up with the best ideas to use metrics to dramatically reduce software vulnerabilities in three to five years. The agenda was as follows:

8:30 am – 9:00 am	Registration
9:00 am – 9:10 am	**Introduction, Safety, Schedule, Charge** Paul E. Black, NIST
9:10 am – 9:15 am	**Federal Cybersecurity Research and Development Strategic Plan** Greg Shannon, White House Office of Science and Technology Policy
9:15 am – 9:30 am	**Opening Remarks** William F. Guthrie, Chief, Statistical Engineering Division, NIST
9:30 am – 10:00 am	**Measuring Software Analyzability** Andrew Walenstein, BlackBerry
10:00 am – 10:30am	**Dealing with Code That is Opaque to Static Analysis** James Kupsch, University of Wisconsin-Madison

10:30 am – 10:50 am	Break
10:50 am – 11:10 am	**Composing Processes for Secure Development Using Process Control Measures** William Nichols, Software Engineering Institute (SEI)
11:10 am – 11:30 am	**Measure Early and Measure Often – SWAMP** Miron Livny, Morgridge Institute for Research
11:30 am – 1:00 pm	Lunch
1:00 pm – 1:20 pm	**CISQ Measures of Secure, Resilient Software** Dr. Bill Curtis, Executive Director, Consortium for IT Software Quality (CISQ)
1:20 pm – 1:40 pm	**Mostly Sunny with a Chance of Cyber-Doom** David Flater, NIST
1:40 pm – 2:00 pm	**Dynamically Proving That Security Issues Exist** Dr. Andrew V. Jones, Vector Software, Inc.
2:00 pm – 2:20 pm	**Breakouts**
2:20 pm – 2:50 pm	Break
2:50 pm – 3:20 pm	**Breakout Reports (6 reports at 5 minutes each)**
3:20 pm – 3:40 pm	**Toward Evidence-Based Low Defect Software Production** James Kirby, Jr., US Naval Research Laboratory
3:40 pm – 4:00 pm	**Using Malware Analysis to Reduce Design Weaknesses** Carol Woody, Ph.D., Software Engineering Institute
4:00 pm – 4:20 pm	**Summary – Our Next Step** Paul E. Black, NIST

Although UL was invited to present based on their position statement, there was no presentation because of illness.

1.3 Non-Measurement Ideas from the Breakout Session

The discussions in the breakout groups were lively. Most of the groups continued their discussions to the end of the break time. Someone from each breakout group took five minutes to report their recommendations to the whole workshop when it reconvened. The workshop was focused on metrics and measures of software as a product and what could be done in a moderate time frame. Although charged to discuss ideas related to the workshop theme, every group included ideas related to software quality, assurance, software development, and cybersecurity in general. This section lists many of those ideas that are outside the scope of the workshop, and thus are not in Section 2. Some came from more than one group.

When we use phrases like “workshop participants” or “some who attended,” we usually mean a group of a dozen or so. In no case were all participants polled and a consensus, or even plurality, determined. Ideas were often brought up by one person, discussed and elaborated by others, then written or reported by yet others. Hence it is difficult to attribute ideas to particular people in most cases. We thank all those who participated in the workshop and made contributions, large and small, to the ideas noted in this report.

1.3.1 Consider Vulnerabilities in All Parts of the Software Life Cycle

Some who attended the workshop thought that security must be designed into the system from the beginning. It must be a part of the requirements of the system and the architecture of the software. Security often touches and influences many pieces of the system, from low-level details such as how data is stored to high-level details such as a global state recording the user's role or whether the user has been authenticated. When security is added later on, it is typically expensive to develop and test, difficult to use, inadequate, or all three [2].

Another caveat is that security cannot just be designed in, then forgotten. Security should be an integrated part of the entire software development lifecycle. Analysts, coders, testers, integrators, and operators all have vital roles into operating a secure system. Security cannot be relegated to a quality hurdle that the development process needs to surmount then forget. If the software development has a separate group of experts who have been thoroughly trained in cybersecurity and in low-vulnerability software, then developing less vulnerable software should be a partnership between the development team and the experts.

1.3.2 Government Contracting and Procurement, Liability, and Insurance

Many workshop participants felt that the Federal Government could lead a significant improvement in software quality by requiring software quality during contracting and procurement and by changing general expectations.

Participants felt that model contract language can include incentives for software to adhere to higher coding and assurance standards or punitive measures for egregious violations of those standards. The defense community [3], the financial sector, the automotive sector and the medical sector have published sample procurement language for cybersecurity and secure software. The focus on the lowest bidder must include provisions for "fitness for purpose" that factor in considerations for secure software. Only products that fulfill technical acceptance requirements should be considered. Software suppliers who have sloppy cyber hygiene should be identified in contract bidding. All software, especially third-party open source software (OSS), should be evaluated to substantiate that it does not have malware or known or new vulnerabilities, as much as feasible. Software should have a "bill of materials" such that those using it could respond to a new threat made public about some component or library in the software.

Participants generally agreed that new exploits will be discovered after software is put into use, hence the need for a bill of materials. They felt that companies developing software should be contractually liable for vulnerabilities discovered after delivery. Such liability clauses might be modeled after those used in the video game industry in the 2000s. Participants did not believe that there should be legal liability at this time. On the other hand, the language of liability clauses needs to be strict enough to, as one participant wrote, "hold companies accountable for sloppy and easily-avoidable errors, flaws, and mistakes."

One complicating factor is that liability includes a concept of responsible party. Responsibility may be hard to determine in the case of "open source" or freely available software.

The Financial Services Sector Coordinating Council (FSSCC) for Critical Infrastructure Protection and Homeland Security produced a 26-page document entitled *Purchasers' Guide to Cyber Insurance Products* defining what cyber insurance is, explaining why organizations need it, describing how it can be procured, and giving other helpful information.

Many software assurance tool agreements include "DeWitt clauses" that prohibit the user from publishing any evaluations or comparisons with other tools. Participants felt that such restrictions slow the development of better techniques and make it difficult for users to determine which tool or tools are most beneficial for them. Contract language could specify an allowance of published evaluations, for example as suggested by Klass and Burger in "Vendor Truth Serum".

1.3.3 Education

Many software developers are not taught the basic principles, practices, and importance of cybersecurity or provided with resources. It was the participants' judgement that educating a large number of programmers in basic cybersecurity practices will significantly reduce vulnerabilities. The Federal Government could fund broad funding of on-line or self-study courses and work with companies to promote widely-available resources.

In addition, software developers should learn how and when to use powerful and sophisticated tools, which are now available. Participants opined that developers need to understand that they shouldn't just turn off red flags raised by tools. As above, many institutions of higher education or training organizations can offer free training, once courses are developed.

Some workshop attendees noted that educating just front-line software developers is not enough. Managers and executives must also be educated in the risk management implications of software vulnerabilities and the importance of investing in cybersecurity and low vulnerability software.

1.3.4 Research Projects for Security, Quality, and Few Vulnerabilities

One participant suggested a major project to provide a single forum where researchers could share samples of code, share findings, collaborate on research, and publish results without intellectual property restrictions. A large, open repository of source code would allow researchers to conduct a wide range of data-driven research. Such research could lead to improved programming practices, ways to spot poor quality or malicious code, and new and improved software security metrics and measures. This must be independent of vendors and model and encourage scientifically valid research. The concept would be similar to the Human Genome Project (HGP), but for software instead of genomes. A critical difference is the intellectual property of software.

Participants felt that there needs to be increased scientifically valid research about the strengths and limitations of software assurance tools. Researchers and users could share their findings through a forum such as suggested above. There might even be a list of verified tools.

Another aspect of such a project is to collect incidences of malware, dead code, and other "code smells" so that they are available to researchers. These could augment Common Vulnerabilities and Exposures (CVE). Along the same lines, participants felt that there should be a repository of computer system breaches, like those mandated by the State of California. Such a repository is analogous to those maintained by the Food and Drug Administration (FDA) for medical device problems and the Federal Aviation Administration (FAA) for aircraft incidents.

Attendees noted that today every vulnerability is addressed with its own special, tactical response. Some suggested that there needs to be a substantial research agenda to develop a science of the appearance, detection, behavior, and useful responses of software vulnerabilities instead of treating each vulnerability as a unique problem with its own measure or metric. Given this knowledge, more generalized capabilities to counter classes of vulnerabilities could be developed. Over time a theory of software vulnerability could be developed that provides a larger context for the problem and systemic measures and metrics for detecting and countering classes of vulnerabilities.

1.3.5 Government Funded Efforts

In the participants' estimation, the Government could fund the research and publication of business cases for secure software, including the cost of security breaches. Such studies or cases would bolster education mentioned in Section 1.3.3.

Workshop attendees suggested that the Federal Government could test or certify the software in widely-used or important modules, libraries, or packages. To partner with the private sector, the Government could fund such testing, perhaps as part of procurement.

Software quality could be improved by following up the Baldridge Cybersecurity Initiative. This may help encourage software companies, according to some participants.

1.3.6 Third Party Review of Software

Some participants felt that software and the software industry should be treated as other industries, like automotive, tobacco, and food. The Government mandated seatbelt use. Could it encourage the software development industry to adopt well-known techniques and practices that industry is reluctant to adopt, because of the belief that such efforts would make them non-competitive? Based on well-established science, the Government could issue directives, create cybersecurity and quality standards, and even mandate compliance. The FDA could enforce standards for medical devices, and the Federal Trade Commission (FTC) could have a role in software apps, since it deals with deceptive advertising [4, 5].

Some participants hoped that the Government would continue to nurture the efforts to add requirements for better security and for lower numbers of vulnerabilities to Federal Information Security Management Act (FISMA) documents, (e.g., NIST Special Publication (SP) SP 800-538, SP 800-64, SP 800-53 and SP 800-53a), and to Department of Defense standards and guidelines.

Participants judged that software could benefit from the programs and criteria of widely-accepted non-governmental organizations. Some possibilities are UL's Cybersecurity Assurance Program (CAP), Consortium for IT Software Quality (CISQ) Code Quality Standards, and Core Infrastructure Initiative (CII) Best Practices badge.

2. Observations and Recommendations

The focus of the workshop was measures and metrics of software as a product. This section details general observations and suggestions of workshop participants. Some participants cautioned that software quality and security metrics may be the wrong emphasis to reduce software vulnerabilities, that such metrics may fade in emphasis as other software metrics have, for example cohesion and McCabe Cyclomatic Complexity.

2.1 Better Code

Participants praised two workshop presentations: Andrew Walenstein's "Measuring Software Analyzability" and James Kupsch's "Dealing with Code that is Opaque to Static Analysis." Both stressed that code should be amenable to automatic analysis. Both presented approaches to define what it means that code is readily analyzed, why analyzability contributes to reduced vulnerabilities, and how analyzability could be measured and increased.

Some participants noted that there are subsets of programming languages that are designed to be analyzable, such as SPARK, or to be less error-prone, like Less Hatton's SaferC. Participants generally favored using better languages, for example, functional languages such as F# or ML. However, there was no particular suggestion of *the* language, or languages, of the future.

Attendees also pointed out that while code-based metrics are important, we can expect complementary results from metrics related to the other aspects of the software. Some aspects are the software architecture and design erosion metrics, linguistic aspects of the code, developers' background, and metrics related to the software requirements.

2.2 More Useful Tool Outputs

There are many powerful and useful software assurance tools available today. No single tool meets all needs. Accordingly, users should use several tools. This is difficult because tools have different output formats and use different terms and classes.

Participants emphasized that tool outputs should be standardized. That is, the more there is common nomenclature, presentation, and detail, the more feasible it is for users to combine tool results with other software assurance information and to choose a combination of tools that is most beneficial for them.

As explained in Section 1.3, participants felt the need for scientifically valid research about tool strengths and limitations, mechanisms to allow publication of third party evaluation of tools, a common forum to share insights about tools, and perhaps even a list of verified or certified tools.

2.3 Security Metrics

Participants didn't note any particular security or vulnerability metrics or measures. However, many participants felt that security or vulnerability measurement (or testing or checking) must be included in *all* phases of software development, as explained in more detail in Section 1.3.1. Except for atypical approaches like Clean Room, this measurement cannot be left as a gate at the end of the production cycle.

2.4 Additional Directions

Some workshop participants were of the opinion that there is a significant need for metrics and measures of binaries or executables. With today's optimizing compilers and with the dependence on many libraries delivered in binary, solely examining source code leaves many avenues for appearance of subtle vulnerabilities.

In the estimation of some attendees, model-based engineering opens the way to writing "source code" at a higher conceptual level and, more importantly, to formal proofs that certain properties are maintained.

2.5 References

[1] Federal Cybersecurity Research and Development Strategic Plan. Available at https://www.whitehouse.gov/sites/whitehouse.gov/files/documents/2016_Federal_Cybersecurity_Research_and_Development_Stratgeic_Plan.pdf

[2] James P. Anderson, "Computer Security Technology Planning Study," October 1972. Available at http://seclab.cs.ucdavis.edu/projects/history/papers/ande72a.pdf

[3] "Suggested Language to Incorporate Software Assurance Department of Defense Contracts," Department of Defense (DoD) Software Assurance (SwA) Community of Practice (CoP) Contract Language Working Group, John R. Marien, chair, and Robert A. Martin, co-chair, February 2016. Available at http://www.acq.osd.mil/se/docs/2016-02-26-SwA-WorkingPapers.pdf Accessed 6 September 2016.

[4] "Mobile Health App Developers: FTC Best Practices," April 2016. Available at http://www.ftc.gov/tips-advice/business-center/guidance/mobile-health-app-developers-ftc-best-practices

[5] Keith Barritt, "3 Lessons: FDA/FTC Enforcement Against Mobile Medical Apps," January 2016. Available at http://www.meddeviceonline.com/doc/lessons-fda-ftc-enforcement-against-mobile-medical-apps-0001

3. Position Statements and Presentations

The program committee invited some of those who submitted position statements to make presentations at the workshop. This section allows those who were invited to publish their position in the manner that they choose. In some cases, this is just the position statement. In other cases, it is an extended version of the position statement. In yet other cases it is the possibly-edited presentation given at the workshop or some combination of all of them.

Please note that the following do not necessarily represent the opinion or result of the National Institute of Standards and Technology (NIST). The appearance here does not imply that NIST endorses any of these ideas or products.

The order here is the original order of presentations planned for the workshop.

3.1 Federal Cybersecurity Research and Development Strategic Plan, Greg Shannon, White House Office of Science and Technology Policy.

3.2 Opening Remarks, William F. Guthrie, Chief, Statistical Engineering Division, NIST.

3.3 Measuring Software Analyzability, Andrew Walenstein, BlackBerry.

3.4 Dealing with Code that is Opaque to Static Analysis, James Kupsch, University of Wisconsin-Madison.

3.5 Ken Modeste, UL.

3.6 Composing processes for secure development using process control measures, William Nichols, Software Engineering Institute.

3.7 CISQ Measures of Secure, Resilient Software, Dr. Bill Curtis, Executive Director, Consortium for IT Software Quality (CISQ).

3.8 Mostly Sunny with a Chance of Cyber, David Flater, NIST.

3.9 Dynamical Proving That Security Issues Exist, Dr. Andrew V. Jones, Vector Software.

3.10 Toward Evidence-Based Low Defect Software Production, James Kirby Jr., US Naval Research Laboratory.

3.11 Using Malware Analysis to Reduce Design Weaknesses, Carol Woody, Ph.D., Software Engineering Institute.

3.12 Measure Early and Measure Often – SWAMP, Miron Livny.