ITL BULLETIN FOR JANUARY 2017

DRAMATICALLY REDUCING SOFTWARE VULNERABILITIES

Paul E. Black, Larry Feldman,[1] and Greg Witte,[1] Editors
Software and Systems Division and Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology (NIST)
U.S. Department of Commerce

Introduction

As the role of technology continues to expand into every aspect of society, finding methods to reduce the number of vulnerabilities in products is more critical than ever. To highlight that need, the *Federal Cybersecurity Research and Development Strategic Plan,*[2] developed under the leadership of the White House Office of Science and Technology Policy (OSTP), makes the case that we could gain valuable protection if the number of vulnerabilities in hardware and software products could be reduced.

These vulnerabilities are often difficult to discover after a product has been developed, and very challenging to correct at that point. Cybersecurity has not kept up with the rapidly accelerating pace of technical innovation and the increasing use of software in society. To put forth some ideas about these challenges, ITL published NIST Interagency Report (NISTIR) 8151, *Dramatically Reducing Software Vulnerabilities: Report to OSTP*. The report presents some technical approaches that have the potential to dramatically help reduce vulnerabilities – by stopping them before they occur, by finding them before they are exploited. and by reducing their impact. The NISTIR is not intended to provide an exhaustive and detailed playbook of software assurance methodologies, but rather provides a concise and consumable set of ideas that can be applied in a reasonably short time to improve protection.

The document's recommendations for reducing vulnerabilities focus on approaches that meet three criteria:

1. Dramatic impact;
2. Three-to-seven-year time frame; and
3. Technical activities.

[1] Larry Feldman and Greg Witte are Guest Researchers from G2, Inc.
[2] The Federal Cybersecurity Research and Development Strategic Plan was released in February 2016 and is available from: https://www.whitehouse.gov/sites/whitehouse.gov/files/documents/2016_Federal_Cybersecurity_Research_and_Development_Stratgeic_Plan.pdf.

Dramatic impact. This area describes approaches that are broadly applicable and that significantly (i.e., up to two orders of magnitude) contribute to the goal of reducing vulnerabilities. The report recognizes that many defects result from simple programming errors. Practical changes in the development approach can significantly reduce the number of these errors, vastly improving the quality of the resulting product. Understanding the specific impact of each approach requires effective methods to measure software quality – such measurement itself is a difficult challenge and one that NIST continues to research and collaborate about with government and industry partners.

Three-to-seven-year time frame. This time frame was selected because it provides sufficient time to implement and measure dramatic changes, based on existing techniques that have not reached their full impact potential. A three-to-seven-year window is reasonable to speculate about; beyond that horizon, it becomes difficult to predict the emergence and influence of new technologies and techniques.

Technical activities. There are varied approaches to reducing software vulnerabilities, many of which are not primarily technical. These approaches cover many aspects of the development life cycle. For example, helping users to meaningfully describe security needs may help to ensure that security is built into the products. Similarly, improving training for those who design, build, test, and use software will help to avoid, detect, and correct product defects. NIST received many ideas (both technical and non-technical) across this broad span. To stay within a manageable scope, the NISTIR focuses on the technical approaches. Non-technical solutions, some of which are described below, are also highly important but are not the focus of the report.

The list of approaches included was based upon a community-based effort, developed over an eight-month period. To support the requested timeline, the focus of the report was kept to the criteria described above, to highlight promising approaches rather than perform a comprehensive analysis. NIST consulted with multiple experts in the software assurance community, including collaboration through OSTP-hosted interagency roundtables; the Software and Supply Chain Assurance (SSCA) Forum; an all-day workshop on Software Measures and Metrics; and a two-day workshop on Reducing Software Defects and Vulnerabilities, hosted by the Software Productivity, Sustainability, and Quality (SPSQ) Working Group of the Networking and Information Technology Research and Development (NITRD) Program.

Technical Approaches

There are many approaches, at varying levels of maturity, which show great promise for reducing the number of vulnerabilities in software. NISTIR 8151 describes some that are reasonably mature and have shown success, so that it is possible to extrapolate into a three-to-seven-year horizon. This list is not exhaustive, but rather is meant to show that it is possible to make significant progress in reducing vulnerabilities and to lay out paths to achieve this ambitious goal. One of the significant themes of the

SPSQ workshop on Reducing Software Defects and Vulnerabilities was the need to improve not just software, but also testing tools through the application of formal techniques.

NISTIR 8151 groups these approaches into five general areas:

1. *Formal Methods*. Formal methods include all software analysis approaches based on mathematics and logic, including parsing, type checking, correctness proofs, model-based development, and correct-by-construction.

2. *System-Level Security.* When software is executed, the system context for the running software defines the resources available to the software, the application program interfaces (APIs) needed to access those resources, and how the software may access (and be accessed by) outside entities. These aspects of a system context may strongly affect the likelihood that software contains vulnerabilities (e.g., complex or buggy APIs increase that likelihood), the feasibility of an attacker exploiting vulnerabilities (e.g., exploitation becomes more feasible if system services are reachable from outside), and the potential impact of an attack (e.g., both damage to system resources and mission-specific costs).

3. *Additive Software Analysis Techniques.* Currently, there are many different tools and techniques—both open source and proprietary software—to analyze software and to check for problems. Additive software analysis refers to a comprehensive approach for enabling the use of multiple advanced software checking tools. The goal of additive software analysis is to foster a continuing accumulation of highly usable analysis modules that add together over time to continually improve the state of the practice in deployed software analysis.

4. *Domain-Specific Software Development Frameworks.* The goal of this approach is to promote the use (and reuse) of well-tested, well-analyzed code, and thus reduce the incidence of exploitable vulnerabilities. A mature domain-specific framework, once learned by software developers, can enable quick production of programs that are well tested both from a software perspective and from a domain knowledge perspective.

5. *Moving Target Defenses (MTD) and Automatic Software Diversity.* The goal of software diversity and MTD is to reduce an attacker's ability to exploit vulnerabilities in software, not to reduce the number of weaknesses in software. This reduction may be achieved by changing the "attack surface," (e.g., adjusting the interface accessible by an attacker, or regenerating system components that have been compromised).

Measures and Metrics

An additional section in NISTIR 8151 deals with measures and evaluations in the broadest sense, including code reviews, software testing, and other techniques. Currently, a significant need exists for precise and rigorous validation measures—even existing methods are only moderately predictive of

software vulnerability. Additional and detailed data is needed to improve the foundation for measurement research.

NISTIR 8151 lists the following three areas of attention:

1. *Encouraging the Use of Measures*. All the extraordinary measures in the world do not help if nobody uses them. Also, nobody *can* act on measures if the measures are not produced and available.

2. *Process Measures*. Gaining an understanding of how to measure the characteristics and effectiveness of development processes will help to identify sources of additional defects (e.g., from developers working overtime) and methods of improvement. Process measures such as hours of effort, number of changes with no acceptance test defects, and acceptance test defect density in delivered code may not have a direct effect on the number of vulnerabilities, but the indirect effects are significant.

3. *Measures of Software as a Product*. This section concentrates on measures that apply to the software itself. Examples include proof of the absence of buffer overflows; assertion of the number of defects per thousand lines of code; and assurance that specifications are met and path coverage achieved by a test suite.

The section distinguishes between base measures and derived measures. A base measure is a simple, basic assessment or count with a clear value. A derived measure, on the other hand, is a function of two or more values of base measures or a mathematical transformation of a base measure. The section also describes several ways to categorize measures.

Non-Technical Approaches

While the technical approaches described are important, the report also highlights the value of non-technical solutions. It illustrates the fact that quality improvement doesn’t have to cost more than traditional approaches. For example, small improvements in planning at the beginning of a project can result in reduced testing and patching costs. History illustrates that significant challenges, such as the need to dramatically reduce vulnerabilities in hardware and software, can often influence demand for improved quality. An example of this model is consumers’ move toward fuel-efficient transportation modes in reaction to increased fuel prices. The demand for reduced vulnerability in technical products could be similarly influenced, such as through more stringent quality requirements in government software acquisition contracts. Increased awareness of the opportunity for reduced vulnerability may result in increased user demand.

While these areas fall outside the scope of the report, they are critical both now and in the future. Similarly, the report does not address research and development that is needed as part of a broader understanding of software and vulnerabilities. Topics such as identifying sources of vulnerabilities, how

vulnerabilities manifest as bugs, and improved scanning during development are also critical, but, again, outside the scope of this report.

At the same time, the report outlines some of the needed steps for moving forward by engaging the broader community, including researchers, funders, developers, managers, and customers/users. NISTIR 8151 addresses: 1) engaging and supporting the research community; 2) education and training; and 3) empowering customers and users of software to meaningfully participate by not only asking for quality, but pushing it.

Conclusion

Through the application of the technical approaches described and continued research into non-technical considerations, NISTIR 8151 provides practical and applicable advice to help achieve the following goals in the next few years:

- Stopping vulnerabilities before they occur, including improved methods for specifying and building software;
- Finding vulnerabilities, including better testing techniques and more efficient use of multiple testing methods; and
- Reducing the impact of vulnerabilities by building architectures that are more resilient, so that vulnerabilities cannot be meaningfully exploited.

NIST looks forward to opportunities to continue fostering research and discussions regarding secure software development, software quality measurement, and additional ways to increase the community demand for improved software quality.

ITL Bulletin Publisher: Elizabeth B. Lennon
Information Technology Laboratory
National Institute of Standards and Technology
elizabeth.lennon@nist.gov