Enhancing the Development Life Cycle To Produce Secure Software

Karen Mercedes Goertzel
Booz Allen Hamilton

Over the past decades, efforts to enhance software development life cycle (SDLC) practices have been shown to improve software quality, reliability, and fault-tolerance. More recently, similar strategies to improve the security of software in organizations such as Microsoft, Oracle, and Motorola have resulted in software products with less vulnerabilities and greater dependability, trustworthiness, and resilience. In its mission to improve the security of software used in America's critical infrastructure and information systems, the Department of Homeland Security's (DHS) Software Assurance Program has sponsored the creation of the book Enhancing the Development Life Cycle to Produce Secure Software*, a source of practical information intended to help developers, integrators, and testers identify and systematically apply security and assurance principles, methodologies, and techniques to current SDLC practices, and thereby increase the security of the software that results. Unlike the numerous other books on secure software development,* Enhancing the Development Life Cycle *does not espouse any specific methodology, process model, or development philosophy. Instead it explains the essentials of what makes software secure, and takes an unbiased look at the numerous security principles and secure development methodologies, practices, techniques, and tools that developers are finding effective for developing secure software – information that readers can leverage in defining their own SDLC security-enhancement strategies.*

In its report to President George W. Bush entitled "Cyber Security: A Crisis of Prioritization" (February 2005), the President's Information Technology Advisory Committee summed up the problem of non-secure software:

> Network connectivity provides "door-to-door" transportation for attackers, but vulnerabilities in the software residing in computers substantially compound the cyber security problem…. Software development is not yet a science or a rigorous discipline, and the development process by and large is not controlled to minimize the vulnerabilities that attackers exploit. Today, as with cancer, vulnerable software can be invaded and modified to cause damage to previously healthy software, and infected software can replicate itself and be carried across networks to cause damage in other systems.
>
> Like cancer, these damaging processes may be invisible to the lay person even though experts recognize that their threat is growing. And as in cancer, both preventive actions and research are critical, the former to minimize damage today and the latter to establish a foundation of knowledge and capabilities that will assist the cyber security professionals of tomorrow to reduce risk and minimize damage for the long term. Vulnerabilities in software that are introduced by mistake or poor practices are a serious problem today. In the future, the nation may face an even more challenging problem as adversaries – both foreign and domestic – become increasingly sophisticated in their ability to insert malicious code into critical software.

Software is considered "secure" when it exhibits three interrelated properties:

1. **Dependability.** The software executes correctly and predictably, even when confronted with malicious or anomalous inputs or stimuli.
2. **Trustworthiness.** The software itself contains no malicious logic or any flaws or anomalies that could be exploited or targeted as vulnerabilities by attackers.
3. **Resilience.** When the software is able to resist most attempted attacks, tolerate the majority of those it cannot resist, and recover with minimal damage from the very few attacks that succeed (i.e., those the software could neither resist nor tolerate).

A number of factors influence how likely software is to consistently exhibit these properties under all conditions. These include:

- The programming language(s), libraries, and development tools used to design, implement, and test the software, and how they were used.
- How the software's re-used, commercial off-the-shelf, and open source components were evaluated, selected, and integrated.
- How the software's executables were distributed, deployed, configured, and sustained.
- The security protections and services provided to the software by its execution environment.
- *The practices used to develop the software, and the principles that governed those practices.*

Experience over the past few decades has shown that enhancing SDLC practices with the objective of improving software quality, reliability, and fault-tolerance does, in fact, result in software that is higher in quality, or more reliable, or more tolerant of faults. More recently, the same SDLC enhancement approach has been applied to improve the *security* of software. By adjusting and, in some cases expanding, SDLC activities to ensure that they consistently adhere to secure specification, design, coding, integration, testing, deployment, and sustainment principles, organizations such as Microsoft, Oracle, Motorola, Praxis High Integrity Systems, and a growing number of others, have been able to report that soon after doing so, they were finding vulnerabilities and weaknesses much earlier in the software's life cycle. In turn, those organizations were able to eradicate problems at a much lower cost than ever before. Moreover, the organizations that institutionalized repeated use of *security-enhanced* SDLC practices found that, over time, the enhanced practices became second-nature to their developers, and fewer and fewer vulnerabilities and weaknesses appeared in their software

in the first place. Not only that, but they also noted that their software's dependability, trustworthiness, and resilience also improved.

Enhancing the Development Life Cycle to Produce Secure Software

This year, the DHS sponsored the revision of its 2006 document, "Security in the Software Life Cycle: Making Development Processes – and Software Produced by Them – More Secure[1]." The new document, retitled "Enhancing the Software Life Cycle to Produce Secure Software," transforms what was essentially a compendium of software security assurance concepts and overviews of methodologies, process models, sound practices (also known as *best practices*), and supporting technologies that, when used, had been reported by their advocates to produce software that is more secure than software built by more traditional methods and tools.

By contrast, the authors of "Enhancing the Software Life Cycle" have transformed their previous survey of the software assurance domain into a source of practical information to enable developers, integrators, and testers to identify and systematically apply security and assurance principles, methodologies, and techniques to enhance their current SDLC practices. The revision's focus has been narrowed and its concept discussions streamlined, while its pragmatic technical content has been expanded and deepened and augmented with extensive lists of references to information (online and in print) on how to implement the various techniques and methodologies described in the document. The new version also reflects pertinent technological, methodological, and philosophical advances that have occurred in the software and software assurance domain since the release of "Security in the Software Life Cycle" more than two years ago.

Several other developments made this revision possible. The Department of Defense (DoD) Technical Information Center (DTIC) sponsored the public release of "Software Security Assurance: A State-of-the-Art Report," which captured and updated much of the survey-type information in "Security in the Software Life Cycle." In addition, DHS produced its drafts of "Software Assurance in Acquisition[2]," and "Practical Measurement Guidance for Software Assurance and Information Security[3]." The DoD drafted "Engineering for System Assurance"[4], the DTIC-sponsored "Project Management for Software Assurance: A State-of-the-Art Report,"[5] and Addison-Wesley published "Software Security Engineering: A Guide for Project Managers" as part of its Software Security Series[6]. Collectively, these publications address many concerns of the secondary intended audience for "Security in the Software Life Cycle," including acquisition managers, project managers, system engineers, and information security practitioners, enabling "Enhancing the Software Development Life Cycle" to focus on its primary audience of developers, integrators, and testers.

Unlike the steadily increasing number of other books on secure software development, secure programming, application security, and software security testing, "Enhancing the Development Life Cycle" strives to remain *methodology/process-agnostic*. Its intent is to explain the essentials and characterize the advantages of a number of security principles, and secure development methodologies, practices, and techniques that have proven effective in the security-enhancement of SDLC activities.

"Enhancing the Development Life Cycle" is currently undergoing a final review by the DHS/DoD co-sponsored Software Assurance Working Groups. Based on comments from those reviewers, the final draft will be produced for public comment late this summer. It will be available for download from the Build Security In Web site at <https://buildsecurityin.us-cert.gov/daisy/bsi/dhs.html>.◆

Notes

1. Goertzel, Karen Mercedes, and Joe Jarzombek. "Security in the Software Life Cycle." CrossTalk. Sept. 2006. Accessed 30 May 2008 at <www.stsc.hill.af.mil/crosstalk/2006/09/0609Jarzombek Goertzel.html>.
2. Accessed 30 May 2008 at <https://buildsecurityin.us-cert.gov/daisy/bsi/908.html?branch=1&language=1>.
3. Accessed 10 June 2008 at <https://buildsecurityin.us-cert.gov/swa/downloads/SwA_Measurement.pdf>.
4. Accessed 30 May 2008 at <www.acq.osd.mil/sse/ssa/docs/SA+Guidebook+v905-22Apr08.pdf>.
5. Abstract accessed 30 May 2008 at <https://www.thedacs.com/techs/abstracts/abstract.php?dan=347617>.
6. Mead, Nancy R., et.al. Software Security Engineering: A Guide for Project Managers. Upper Saddle River, NJ: Addison-Wesley, 2008.

About the Author

Karen Mercedes Goertzel, Certified Information Systems Security Professional, is a subject-matter expert in software assurance, cyber security, and information assurance. She supports the DHS' software assurance program, not least as lead author/editor of "Security in the Software Life Cycle." Goertzel coordinated the Information Assurance Technology Analysis Center (IATAC)/ Data and Analysis Center for Software "Software Security Assurance: A State-of-the-Art Report," as well as IATAC's "The Insider Threat to Information Systems: A State-of-the-Art-Report." She was a contributing author to the National Institute of Standards and Technology Special Publication 800-95, "Guide to Secure Web Services," editor/contributing author of the National Security Agency's "Guidance for Addressing Malicious Code Risk," and has been published previously in CrossTalk. As lead technologist for the Defense Information System Agency's Application Security Project from 2002-2004, she was the leading contributor to the DoD Application Security Developer's Guides upon which the Defense Information System Agency's Application Security and Development Security Technical Implementation Guide is (in large part) based. Goertzel has presented at numerous conferences and workshops, including the American Institute of Engineers' 2008 Military Open Source Software Conference and the DoD's 2008 System and Software Technologies Conference. Before joining Booz Allen Hamilton, she was a specialist in the specification and architecture of high assurance cross-domain information sharing solutions for defense and civil establishments in the U.S., Canada, Australia, and NATO.

Booz Allen Hamilton
8283 Greensboro DR
H5061
McLean, VA 22102
Phone: (703) 902-6981
Fax: (703) 902-3537
E-mail: goertzel_karen@bah.com