A Look at "Software Security Engineering: A Guide for Project Managers"©

Julia H. Allen, Dr. Robert J. Ellison, and Dr. Nancy R. Mead
SEI

Sean Barnum and Dr. Gary McGraw
Cigital, Inc.

The goal of software security engineering is to build better, defect-free software. The book "Software Security Engineering: A Guide for Project Managers" [1][1]*—and its key resource, the Build Security In (BSI) Web site—provide software project managers with sound practices that they can evaluate and selectively adopt to help reshape their own development practices. Software developed and assembled using these practices should contain significantly fewer exploitable weaknesses.*

Software is ubiquitous. Many of the products, services, and processes that organizations use and offer are highly dependent on software to handle the sensitive and high-value data on which people's privacy, livelihoods, and very lives depend. National security relies on increasingly complex, interconnected, software-intensive information systems—systems that (in many cases) use the Internet or Internet-exposed private networks as their means for communication and transporting data.

Dependence on IT makes software security a key element of business continuity, disaster recovery, incident response, and national security. Software vulnerabilities can jeopardize intellectual property, consumer trust, business operations and services, and a broad spectrum of critical applications and infrastructures, including everything from process control systems to commercial application products.

The integrity of critical digital assets (systems, networks, applications, and information) depends on the reliability and security of the software that enables and controls those assets. However, business leaders and informed consumers have growing concerns about the scarcity of practitioners with requisite competencies to address software security [2]. They have concerns about suppliers' capabilities to build and deliver secure software that can be used with confidence and without fear of compromise. Application software is the primary gateway to sensitive information. According to the Deloitte survey of 169 major global financial institutions, current application software countermeasures are no longer adequate—application security is the number one issue for chief information officers [3].

The absence of security discipline in today's software development practices often produces software with exploitable weaknesses. Security-enhanced processes and practices—and the skilled people to manage and perform them—are required to build software that can be trusted to operate more securely than the software being used today.

That being said, there is an economic counterargument, or at least the perception of one. Some business leaders and project managers believe that developing secure software slows the process and adds to the cost while not offering any apparent advantage. In many cases, when the decision reduces to *ship now* or *be secure and ship later*, ship now is almost always the choice made by those who control the money but have no idea of the risks. Information combating this argument—showing ways software security has led to cost and schedule reduction and documented successful experiences (e.g., Microsoft's Security Development Lifecycle)—is out there.

> *"Security-enhanced processes and practices ... are required to build software that can be trusted to operate more securely than the software being used today."*

The Goal of Software Security Engineering

Software security engineering is using practices, processes, tools, and techniques for addressing issues in every phase of the software development life cycle (SDLC). Software that is developed with security in mind is typically more resistant to both intentional attack and unintentional failures. One view of secure software is that software is engineered "so that it continues to function correctly under malicious attack" [4] and is able to recognize, resist, tolerate, and recover from events that intentionally threaten its dependability. Broader views that can overlap with software security (e.g., software safety, reliability, and fault tolerance) include proper functioning in the face of unintentional failures or accidents, inadvertent misuse and abuse, and software defect and weakness reduction to the greatest extent possible (regardless of its cause).

The goal of software security engineering is to build better, defect-free software. Software-intensive systems that are constructed using more securely developed software are better able to:

- Continue operating correctly in the presence of most attacks by either resisting the exploitation of weaknesses in the software or tolerating the failures that result from such exploits.
- Limit the damage from attack-triggered fault failures that the software was unable to resist—or tolerate and recover quickly from those failures.

Software Security Practices

No single practice offers a universal silver bullet for software security. With this in mind, "Software Security Engineering" provides software project managers with sound practices that they can evaluate and selectively adopt to help reshape their own development practices. The objective is to increase the security and dependability of the software produced by these practices, both during its development and its operation.

The book—and material referenced on the BSI Web site at <https://buildsecurityin.us-cert.gov>—identify and compare potential new practices that can be adapted to augment a project's current software development practices. These resources also greatly increase the likeli-

hood of producing more secure software and meeting specified security requirements. As one example, assurance cases can be used to assert and specify desired security properties, including the extent to which security practices have been successful in satisfying security requirements.

Software developed and assembled using software security practices should contain significantly fewer exploitable weaknesses. Such software can be relied on to more capably recognize, resist or tolerate, and recover from attacks—in turn functioning more securely in an operational environment. Project managers responsible for ensuring that software and systems adequately address their security requirements throughout the SDLC can review, select, and tailor guidance from the book and Web site as part of normal project management activities.

The five key takeaways from the book are as follows:

1. Software security is about more than eliminating vulnerabilities and conducting penetration tests. Project managers need to take a systematic approach to incorporate sound software security practices into their development processes. Examples include security requirements elicitation, attack pattern and misuse/abuse case definition, architectural risk analysis, secure coding and code analysis, and risk-based security testing.
2. Network security mechanisms and IT infrastructure security services do not sufficiently protect application software from security risks.
3. Software security initiatives should follow a risk management approach to identify priorities and what is good enough, understanding that software security risks will change throughout the life cycle. Risk management reviews and actions are conducted during each SDLC phase.
4. Developing secure software depends on understanding the operational context in which it will be used. This context includes conducting end-to-end analysis of cross-system work processes, working to contain and recover from failures using lessons learned from business continuity, and exploring failure analysis and mitigation to deal with system and system-of-systems complexity.
5. Project managers and software engineers need to think like an attacker in order to address the range of things that software should not do and how software can better resist, tolerate, and recover when under attack. The use of attack patterns and misuse/abuse cases throughout the SDLC encourages this perspective.

Practice Maturity and Relevance

As a community, we recognize that some software security practices are in broader use and thus more tested and mature than others, such as security coding practices and vulnerability testing. As a practice description and selection aid, descriptive tags mark the book's sections and key practices in two practical ways:

1. Identifying the content's relative *maturity of practice* as follows:
   - **Maturity Level 1 (L1):** The content provides guidance for how to think about a topic for which there is no proven or widely accepted approach. The intent of the description is to raise awareness and aid in thinking about the problem and candidate solutions. The content may also describe promising research results that may have been demonstrated in a constrained setting.
   - **Maturity Level 2 (L2):** The content describes practices that are in early pilot use and are demonstrating some successful results.
   - **Maturity Level 3 (L3):** The content describes practices that are in limited use in industry or government organizations, perhaps for a particular market sector.
   - **Maturity Level 4 (L4):** The content describes practices that have been successfully deployed and are in widespread use. These practices can be used with confidence. Experience reports and case studies are typically available.
2. Identifying the designated audiences for which each chapter section or practice is most relevant:
   - **E:** Executive and senior managers.
   - **M:** Project and mid-level managers.
   - **L:** Technical leaders, engineering managers, first-line managers, and supervisors.

> *"Software developed and assembled using software security practices should contain significantly fewer exploitable weaknesses ..."*

Build Security In: A Key Resource

Since 2004, the DHS Assurance Program has sponsored development for the BSI Web site, a significant resource used in developing "Software Security Engineering." BSI content, referenced throughout the book, is based on the principle that software security is fundamentally a software engineering problem and must be managed in a systematic way throughout the SDLC.

BSI both contains and links to a broad range of information about sound practices, tools, guidelines, rules, principles, and other knowledge to help project managers deploy software security practices and build secure and reliable software. Contributing authors to this book and articles appearing on the BSI site include senior staff from the SEI and Cigital, Inc.

Readers can consult BSI for additional details, ongoing research results, and information about related Web sites, books, and articles.

Start the Journey

As software and security professionals, we will never be able to get ahead of the game by addressing security solely as an operational issue. Attackers are creative, ingenious, and increasingly motivated by financial gain. They have been learning how to exploit software for several decades; the same is not true for software engineers, and we need to change this. Given the extent to which nations, economies, businesses, and families rely on software to sustain and improve the quality of life, we must make significant progress in putting higher quality and more secure software into production. The practices described in "Software Security Engineering" serve as a useful starting point.

Each project manager needs to carefully consider the knowledge, skills, and competencies of their development team, their organizational culture's tolerance (and attention span) for change, and the degree to which sponsoring executives have bought in (a prerequisite for sustaining any improvement initiative). In some cases, it may be best to start with secure software coding and testing practices: They are the most mature, have a fair level of automated support, and can demonstrate some early successes, providing visible benefits in helping software security efforts gain sup-

port and build momentum. On the other hand, secure software requirements engineering and architecture and design practices offer opportunities to address more substantive root cause issues early in the life cycle that, if left unaddressed, will show up in the code and test phase. Practice selection and tailoring are specific to each organization and project based on objectives, constraints, and the criticality of the software under development.

Project managers and software engineers need to develop a better understanding of what constitutes secure software—honing their skills to think like an attacker—applying this mindset throughout the SDLC. The book describes practices to get this ball rolling, such as attack patterns and assurance cases. Alternatively, if you have access to experienced security analysts, adding a few of them to your development team can get this jump-started.

Two of the key project management practices are 1) defining and deploying a risk management framework to help inform practice selection and determine where best to devote scarce resources and 2) identifying the best way to integrate software security practices into the organization's current SDLC.

Also keep in mind that this process, if done properly, will take time. As John Steven stated:

> Don't demand teams to begin conducting every activity on day one. Slowly introduce the simplest activities first, then iterate ... [Have] patience. It will take at least three to five years to create a working, evolving software security machine. Initial organization-wide successes can be shown within a year. Use that time to obtain more buy-in and a bigger budget. [5]

Clearly, there is no one-size-fits-all approach. Project managers and their teams need to think through the choices, define their tradeoff and decision criteria, learn as they go, and understand that this effort requires continuous refinement and improvement.

In Closing

Sound software security engineering practices should be incorporated throughout the entire SDLC. "Software Security Engineering" is one resource that captures both standard and emerging software security practices and explains why they are needed to develop more security-responsive and robust systems.◆

Software Defense Application

The book "Software Security Engineering: A Guide for Project Managers"—based primarily around the DHS' Build Security In Web site—is a defense software industry mainstay for selecting the right systems assurance tools. This article illuminates ways to use the book in planning software security improvements, from the early development stages through deployment and operations. As well, software developed using the suggested practices will result in on-time and on-budget projects that are more predictably secure.

References

1. Allen, Julia, et al. *Software Security Engineering: A Guide for Project Managers*. Upper Saddle River, NJ: Addison-Wesley, 2008.
2. Carey, Allan. "2006 Global Information Security Workforce Study." IDC. Oct. 2006 <www.isc2.org/uploadedFiles/Industry_Resources/workforcestudy06(1).pdf>.
3. Deloitte. *2007 Global Security Survey: The Shifting Security Paradigm*. Sept. 2007 <www.deloitte.com/assets/Dcom-Serbia/Local%20Assets/Documents/rs_Deloitte_Global_Security_Survey_2007.pdf>.
4. McGraw, Gary. *Software Security: Building Security In*. Boston: Addison-Wesley Professional, 2006.
5. Steven, John. "Adopting an Enterprise Software Security Framework." *IEEE Security & Privacy 4.2* (Mar./Apr. 2006): 85-87 <https://buildsecurityin.us-cert.gov/daisy/bsi/resources/published/series/bsi-ieee/568. html>.

Note

1. Material from this article has been taken from the preface and Chapter 8 of "Software Security Engineering: A Guide for Project Managers." It is reproduced with permission of Pearson Education, Inc. For additional information about the book, including a full table of contents, please refer to <www.informit.com/store/product.aspx?isbn=032150917X> and <www.sei.cmu.edu/library/abstracts/books/032150917X.cfm>. As well, podcasts including Julia Allen, Nancy Mead, and Gary McGraw are a nice introduction to the book's content. Find the CERT Podcast series at <www.cert.org/podcast/#softsecurity>.

Coming Events

March 9-12
12th Semi-Annual Software Assurance Forum
McLean, VA
https://buildsecurityin.us-cert.gov/daisy/bsi/events.html

April 26-29
22nd Annual Systems and Software Technology Conference
Salt Lake City, UT
www.sstc-online.org

May 3-7
DISA Customer Partnership Conference 2010/AFCEA Technology Showcase
Nashville, TN
http://events.jspargo.com/disa10

May 10-14
PSQT 2010 West
Las Vegas, NV
www.psqtconference.com/2010west

May 24-27
Siemens PLM Connection 2010
Nashville, TN
http://event.plmworld.org

June 6-10
IBM Rational Software Conference
Orlando, FL
http://www-01.ibm.com/software/rational/innovate

June 6-11
Better Software Conference
Las Vegas, NV
www.sqe.com/BetterSoftwareConf

Coming Events: **Please submit coming events that are of interest to our readers at least 90 days before registration. E-mail announcements to: <marek.steed.ctr@hill.af.mil>.**

About the Authors

Julia H. Allen is a senior member of the technical staff within the CERT Program at the SEI. In addition to her work in software security and assurance, Allen conducts research in security governance, operational resilience, and metrics. She is the author of "The CERT Guide to System and Network Security Practices," and "Governing for Enterprise Security," and co-hosts discussions for the CERT Podcast Series: "Security for Business Leaders."

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-3890
Phone: (412) 268-7700
E-mail: jha@sei.cmu.edu

Robert J. Ellison, Ph.D., is a member of the Survivable Systems Engineering Team within the CERT Program at the SEI, and has served in a number of technical and management roles. Ellison regularly participates in the evaluation of software architectures and contributes from the perspective of security and reliability measures. His research draws on that experience to integrate security issues into the overall architecture design process. Ellison's current work explores developing reasoning frameworks to help architects select and refine design tactics to mitigate the impact of a class of cyberattacks.

E-mail: ellison@sei.cmu.edu

Nancy R. Mead, Ph.D., is a senior member of the technical staff in the Survivable Systems Engineering Group within the CERT Program at the SEI. She is a faculty member in the Master of Software Engineering and Master of Information Systems Management programs at Carnegie Mellon University. Mead is a fellow for the IEEE and the IEEE Computer Society and is also a distinguished member of the Association for Computing Machinery. Mead has more than 100 publications and invited presentations.

E-mail: nrm@sei.cmu.edu

Sean Barnum is a principal consultant at Cigital and is technical lead for their federal services practice. He has more than 20 years of experience in the software industry in the areas of development, software quality assurance, quality management, process architecture and improvement, knowledge management, and security. He is involved in numerous knowledge standards-defining efforts, including Common Weakness Enumeration, Common Attack Pattern Enumeration and Classification, and other elements of software assurance programs for the DHS and DoD. He is also the lead technical subject matter expert for the Air Force Application Software Assurance Center of Excellence.

Cigital, Inc.
21351 Ridgetop CR, STE 400
Dulles, VA 20166
Phone: (703) 473-8262
E-mail: sbarnum@cigital.com

Gary McGraw, Ph.D., is the chief technical officer of Cigital, Inc. He is a globally recognized authority on software security and the author or co-author of six best-selling books on this topic. The latest, "Exploiting Online Games," was released in 2007. His other titles include "Java Security," "Building Secure Software," "Exploiting Software," and "Software Security"; he is also contributing editor for the Addison-Wesley Software Security series. McGraw has written more than 90 peer-reviewed scientific publications, and authors a monthly column for Dark Reading <www.darkreading.com>.

Phone: (703) 404-9293
E-mail: gem@cigital.com