This article is part of the IEEE Reliability Society 2008 Annual Technology Report.

Software Security Engineering: A Key Discipline for Project Managers

Julia H. Allen
Software Engineering Institute (SEI)
Email: jha@sei.cmu.edu

Sean Barnum
Cigital

Robert J. Ellison
SEI

Gary McGraw
Cigital

Nancy R. Mead
SEI.

The goal of software security engineering is to build better, defect-free software Resources are now available that 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 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—and by extension citizens' personal safety—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 information technology 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. And in today's operational environment software is under an increasing quantity and complexity of attack.

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 [1] They have concerns about suppliers' capabilities to build and deliver secure software that they can use 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,

2007 Global Security Survey: The Shifting Security Paradigm [2], current application software countermeasures are no longer adequate. In the survey, Gartner identifies application security as the number one issue for chief information officers (CIOs).

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 them and perform them—are required to build software that can be trusted to operate more securely than software being used today. Bridging the capability gap will require addressing today's shortage in both practitioners skilled in how to execute software security practices and managers who have the ability to understand, select and direct their application.

That said, there is an economic counter-argument, 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 to combat this argument, including how software security can potentially reduce cost and schedule, is becoming available based on earlier work in software quality and the benefits of detecting software defects early in the life cycle along with documented experiences such as Microsoft's Security Development Lifecycle.

The Goal of Software Security Engineering

To address these challenges effectively, it is important that software development leaders are familiar with and competent in the discipline of software security engineering. Software security engineering is using practices, processes, tools, and techniques that enable you to address security 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 software that is engineered "so that it continues to function correctly under malicious attack" [3] and is able to recognize, resist, tolerate, and recover from events that intentionally threaten its dependability. Broader views that can overlap with software security (for example, software safety, reliability, and fault tolerance) include proper functioning in the face of unintentional failures or accidents and inadvertent misuse and abuse, as well as reducing software defects and weaknesses to the greatest extent possible regardless of their 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 by attackers or *tolerating* the failures that result from such exploits

- limit the damage resulting from any failures caused by attack-triggered faults that the software was unable to resist or tolerate and recover as quickly as possible from those failures

Software Security Practices

No single practice offers a universal silver bullet for software security. Software security engineering provides software project managers with a variety of sound practices and resources

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.

It is the responsibility of the software development manager to leverage available guidance in the identification and comparison of potential new practices that can be adapted to augment a project's current software development practices, greatly increasing the likelihood 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 then be relied on to more capably recognize, resist or tolerate, and recover from attacks and thus function 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 resources such as the Build Security In Web site and the recently published book *Software Security Engineering: A Guide for Project Managers.*

Five key principles of software security engineering 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 the 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 development lifecycle. Risk management reviews and actions are conducted during each phase of the SDLC.

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 system complexity.

5. Project managers and software engineers need to learn 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.

Two Key Resources

In May 2008, Addison-Wesley published the book *Software Security Engineering: A Guide for Project Managers* under both their Software Security Series and their SEI Series in Software

Engineering. This book by the authors of this article explores software security engineering from the project manager's perspective offering valuable contextual explanations for software security, descriptions of a varied set of potential practices and resources available, and guidance for selecting and deploying them appropriately. This book is an excellent starting point and referential resource for software development leaders looking to get a handle on software security.

Since 2004, the U.S. Department of Homeland Security Software Assurance Program has sponsored development for the Build Security In (BSI) Web site (https://buildsecurityin.us-cert.gov/), which is one of the significant resources used in developing *Software Security Engineering*. BSI content 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 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 *Software Security Engineering* and the articles appearing on the BSI site include senior staff from the Carnegie Mellon® Software Engineering Institute (SEI) and Cigital, Inc., as well as other experienced software and security professionals.

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 our nations, our economies, our businesses, and our families rely on software to sustain and improve our 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* and on the Build Security In website 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 given that these are the most mature, have a fair level of automated support, and can demonstrate some early successes, providing visible benefit to help software security efforts gain support 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 code and test. 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 better understand what constitutes secure software and develop their skills to think like an attacker so this mindset can be applied throughout the SDLC. The above resources describe 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 how best to integrate software security practices into the organization's current software development life cycle.

John Steven states [4]

> "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."

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.

References

[1] Carey, Allan. "2006 Global Information Security Workforce Study." Framingham, MA: IDC, 2006. https://www.isc2.org/download/workforcestudy06.pdf

[2] Deloitte Touche Tohmatsu. *2007 Global Security Survey: The Shifting Security Paradigm.* September 2007. http://www.deloitte.com/dtt/cda/doc/content/dtt_gfsi_GlobalSecuritySurvey_20070901(1).pdf

[3] McGraw, Gary. *Software Security: Building Security In*. Boston, MA: Addison-Wesley Professional, 2006 (ISBN 0-321-35670-5).

[4] Steven, John. "Adopting an Enterprise Software Security Framework." *IEEE Security & Privacy 4*, 2 (March/April 2006): 84–87. https://buildsecurityin.us-cert.gov/daisy/bsi/resources/published/series/bsi-ieee/568.html

[5] Allen, Julia; Barnum, Sean; Ellison, Robert; McGraw, Gary; Mead, Nancy. *Software Security Engineering: A Guide for Project Managers*, Addison-Wesley, 2008.

[6] Allen, Julia & Pollak, William. "Security Is A Software Issue." CERT Podcast Series: Security for Business Leaders, May 2008. http://www.cert.org/podcast/show/20080527allen.html

The Build Security In website is located at:
https://buildsecurityin.us-cert.gov

For additional information about the book *Software Security Engineering*, including a full table of contents, please refer to:
http://www.softwaresecurityengineering.com
http://www.sei.cmu.edu/publications/books/cert/software-security-engineering.html
http://www.informit.com/store/product.aspx?isbn=032150917X