Chapter 2
A Survey on Secure Software Development Lifecycles

José Fonseca
DEI/CISUC, University of Coimbra/UDI, Polytechnic Institute of Guarda, Portugal

Marco Vieira
DEI/CISUC, University of Coimbra, Portugal

ABSTRACT

This chapter presents a survey on the most relevant software development practices that are used nowadays to build software products for the web, with security built in. It starts by presenting three of the most relevant Secure Software Development Lifecycles, which are complete solutions that can be adopted by development companies: the CLASP, the Microsoft Secure Development Lifecycle, and the Software Security Touchpoints. However it is not always feasible to change ongoing projects or replace the methodology in place. So, this chapter also discusses other relevant initiatives that can be integrated into existing development practices, which can be used to build and maintain safer software products: the OpenSAMM, the BSIMM, the SAFECode, and the Securosis. The main features of these security development proposals are also compared according to their highlights and the goals of the target software product.

INTRODUCTION

The Software Development Lifecycle (SDL) is a conceptual model used by software houses in the management of the process of analyzing, developing, controlling and maintaining software (Sommerville, 2010). Some of the most well-known models are the Waterfall (Royce, 1970), the Rapid Application Development (Martin, 1991) and the Spiral (Boehm, 1986). At the time when these SDLSs were developed, the software security awareness was not as relevant as it is today, so it was not a big concern to take into account. In fact, the typical approach of dealing only with development best practices is not sufficient for current applications that have to face the constant pressure of web attacks, although they can improve the overall quality and help mitigate some common issues.

DOI: 10.4018/978-1-4666-4301-7.ch002

Copyright © 2014, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

These traditional SDLs are still in widespread use nowadays, but they are not effective when building secure systems that have to face the huge number of threats that can arise from anywhere, like those that come from the web and are so pervasive (Howard & LeBlanc, 2003). Both logic and coding bugs must be thoroughly addressed during all the phases of the development process, therefore reducing the cost of deploying unsecure application. This is of utmost importance for web applications that will be exposed to the growing number of hackers and organized crime that can strike at any time, from any place in the Globe. This is what an integrated Secure Software Development Lifecycles (SSDL) does from the start to the end of the life of an application. In fact, using a SSDL is one of the recommendations of the Verizon's 2009 data breach report in order to prevent the application layer type of attacks, including SQL Injection and XSS (Baker et al., 2009).

This chapter presents an overview of the most important SSDLs that are used nowadays to build software products that have to face the many threats that come from the web: the Open Web Application Security Project (OWASP) Comprehensive, Lightweight Application Security Process (CLASP), the Microsoft Secure Development Lifecycle, and the Software Security Touchpoints. Although there is a general consensus about the advantages of using a SSDL, this subject is still in its early adoption by the industry. It takes time to implement and execute, it costs money and it implies a change in the way organization works, which is usually difficult to achieve. The way a secure software should be developed is still generating a growing number of discussions and there is a considerable number of proposals trying to gain adopters and overcome the problems and technical difficulties of applying them in the real world (Higgins, 2009). This chapter also introduces other relevant initiatives, which can be adapted to the existing SDL, devoted to building and maintaining a safer software product: the Open Software Assurance Maturity Model (OpenSAMM), the Building Security In Maturity Model (BSIMM), the Software Assurance Forum for Excellence in Code (SAFECode) and the Securosis building a web application security program.

This chapter also discusses the issue of selecting a software development lifecycle according to the reality of the software product being developed. This involves identifying the security issues that should be addressed from a development point-of-view and then map these issues with the features of existing lifecycles to make the right choice and tune any relevant aspects.

SOFTWARE DEVELOPMENT AND SECURITY

One important metric of software quality is assurance: "a level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its lifecycle, and that the software functions in the intended manner" (CNSS Secretariat, 2006). To achieve software assurance developers need to build assured software: "Software that has been designed, developed, analyzed and tested using processes, tools, and techniques that establish a level of confidence in its trustworthiness appropriate for its intended use" (CNSS Secretariat, 2006). To achieve this goal, developers must rethink the software development process and address all the phases of the SDL: design, code and documentation (Howard & LeBlanc, 2003). This is like applying the defense-in-depth strategy to the various phases of the software development lifecycle making it more security aware.

To understand the security measures that vendors use for software assurance, Jeremy Epstein analyzed eight software vendors with small to very large revenues (Epstein, 2009). The security measures analyzed were software developer training, software design review, execution of penetration testing using humans and tools dur-

15 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/survey-secure-software-development-lifecycles/77697

Related Content

Cooperation Patterns and Adaptation Patterns for Service-Based Inter-Organizational Workflows
Boukhedouma Saida, Oussalah Mourad, Alimazighi Zaiaand Tamzalit Dalila (2014). *Uncovering Essential Software Artifacts through Business Process Archeology (pp. 250-283).*
www.irma-international.org/chapter/cooperation-patterns-and-adaptation-patterns-for-service-based-inter-organizational-workflows/96624

A Novel Approach to Parkinson's Disease Progression Evaluation Using Convolutional Neural Networks
Mhamed Zineddine (2023). *International Journal of Software Innovation (pp. 1-26).*
www.irma-international.org/article/a-novel-approach-to-parkinsons-disease-progression-evaluation-using-convolutional-neural-networks/315655

A Framework for Analyzing Structural Mechanisms Deployed to Support Traditional and Agile Methods: Making Sense of “Democratization” in the Software Development Workplace
Michal Dolezeland Alena Buchalcevova (2021). *Balancing Agile and Disciplined Engineering and Management Approaches for IT Services and Software Products (pp. 205-227).*
www.irma-international.org/chapter/a-framework-for-analyzing-structural-mechanisms-deployed-to-support-traditional-and-agile-methods/259179

A Method for Detecting Bad Smells and its Application to Software Engineering Education
Yuki Ito, Atsuo Hazeyama, Yasuhiko Morimoto, Hiroaki Kaminaga, Shoichi Nakamuraand Youzou Miyadera (2015). *International Journal of Software Innovation (pp. 13-23).*
www.irma-international.org/article/a-method-for-detecting-bad-smells-and-its-application-to-software-engineering-education/122789

Achieving Effective Health Information Systems
Jim Warren, Karen Dayand Martin Orr (2009). *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications (pp. 560-583).*
www.irma-international.org/chapter/achieving-effective-health-information-systems/21088