Building Secure Software Using XP

Walid Al-Ahmad, King Saud University, Saudi Arabia

ABSTRACT

Security is an important and challenging aspect that needs to be considered at an early stage during software development. Traditional software development methodologies do not deal with security issues and so there is no structured guidance for security design and development; security is usually an afterthought activity. This paper discusses the integration of XP with security activities based on the CLASP (Comprehensive Lightweight Application Security Process) methodology. This integration will help developers using XP develop secure software by applying security measures in all phases and activities, thereby minimizing the security vulnerabilities exploited by attackers.

Keywords: *Agile Methods, Comprehensive Lightweight Application Security Process (CLASP), Secure Software Engineering, Security Best Practices, XP*

INTRODUCTION

Software attacks are possible because software systems contain vulnerabilities in architecture, design, and implementation. According to the Computer Emergency Response Team (CERT), the number of vulnerabilities continues to increase. The total number of vulnerabilities cataloged in the year 2004 was 3,780 while in 2006 the approximate number was 8,064, which indicates an increase of 113% (CERT, 2011). According to another source (NVD, 2011), the National Vulnerability Database, the number of vulnerabilities reported in 2006 was 6,608 while in 2009 the number was 7,171.

Security is not a feature that can just be added on to a software system. This is the reason why more and more organizations are making software security a priority. Due to the increasing frequency and sophistication of malicious attacks against software systems, mainstream software development methodologies must include security as one of the main objectives. Security should be integrated into all activities of a software development methodology. A number of researchers have recently recognized the need for security to be integrated into the Software Development Lifecycle (SDLC) (Aderemi & Seok-Won, 2010; DHS, 2011; Ge et al., 2006; Jones & Rastogi, 2004; Nicolaysen et al., 2010). The importance of building secure software has also been recognized by many international standardization and governments agencies such as ISO 27001 (International Organization for Standardization, 2005), NIST (Kissel et al., 2008), the Department of Homeland Security (DHS, 2011), among others.

DOI: 10.4018/jsse.2011070104

Agile processes are of increasing interest in software development, most significantly in web applications. IT projects may fail due to many reasons. One of the root causes for IT projects failure is related to requirements. Software projects developed by programmers who start programming without detailed understanding of requirements (including security requirements) and design can create chaos and cause the failure of the project. eXtreme Programming (XP) is an agile and flexible software development methodology that has smaller iterations and accepts changing requirements (XP, 2011). It is the most documented and widely used agile software development methodology. As is the case with all agile software development methodologies, XP does not provide support for security in a systematic way. A study carried out by Nicolaysen et al. (2010), focusing on how information security is addressed in an agile context, has indicated that most agile software development organizations do not use any particular methodology to achieve security goals. According to the study, security issues must be addressed adequately during an agile software development process. The reasons for neglecting security issues in software development efforts are well-explained in Jones and Rastogo (2004). The main objective of this research work is to fill in this gap by integrating security into XP in a systematic way while preserving its agility. The contribution of this research work is not the development of a new method or process that addresses security concerns. Rather, the research investigates the XP development method and the structured and comprehensive CLASP security method in order to integrate them to address the development of secure software.

CLASP provides a structured way to concentrate on security issues throughout the software development lifecycle (SDLC) (Viega, 2005). It has been developed by the Open Web Application Security Project (OWASP) which is a non-for-profit organization focused on improving security of applications (http://www.owasp.org/). CLASP is process-oriented and can fit into traditional models such as the waterfall as well as iterative such as IBM Rational Unified Process (RUP) and XP. In fact, a CLASP plug-in has already been implemented to add security to the IBM RUP software development framework (IBM, 2011).

This paper discusses the integration of CLASP security methodology into the well-known XP agile software development methodology. This will help developers build more secure software using the XP method. Our approach to extend XP with security uses the XP practices to complement them with CLASP security activities.

This paper is structured as follows: First, the basics of XP are briefly described with a focus on XP key practices that will be targets for integration with security activities. Next, the article describes the best practices of the CLASP methodology that form the cornerstones of the integration process. Further, the reasons why CLASP has been used to fully integrate security into the XP methodology are explained. The approach to integrate CLASP into XP is then presented and discussed. Finally, the article presents some conclusions and provides glimpses of future work.

EXTREME PROGRAMMING (XP)

Extreme Programming (XP) is one of the most widely used agile software development methodologies. Agile methodologies are those capable of adapting to changing requirements unlike the traditional methodologies such as the Waterfall model where requirements have to be fully understood before development (XP, 2011). XP deals well with changing requirements through an iterative lifecycle with shorter cycles. XP has four main activities in its lifecycle: coding, testing, listening and designing (Paulk, 2001). Each of these activities involves programmers, customers, mangers, as equal partners. It focuses on customer satisfaction by allowing customers to work hand-in-hand with developers in requirement identification, testing the program and adding to the system according to the new requirements. The team works