Chapter 24
Meta-Modeling Based Secure Software Development Processes

Mehrez Essafi
University of Manouba, Tunisia

Henda Ben Ghezala
University of Manouba, Tunisia

ABSTRACT

This work suggests a multilevel support to software developers, who often lack knowledge and skills on how to proceed to develop secure software. In fact, developing software with such quality is a hard and complex task that involves many additional security-dedicated activities which are usually omitted in traditional software development lifecycles or integrated but not efficiently and appropriately deployed in some others. To federate all these software security-assurance activities in a structured way and provide the required guidelines for choosing and using them in a flexible development process, authors used meta-modeling techniques and dynamic process execution that consider developer's affinities and product's states. The proposed approach formalizes existing secure software development processes, allows integration of new ones, prevents ad-hoc executions and is supported by a tool to facilitate its deployment. A case study is given here to exemplify the proposed approach application and to illustrate some of its advantages.

1. INTRODUCTION

Software is everywhere (McGraw, 2006). It sometimes manipulates sensitive data and resources, that may interest attackers, making its' security becoming not only a product quality factor but also, a critical requirement that should be considered at least at a minimum level in all parts of any application systems (Hein, 2009).

Aware of the gravity of the situation, attackers are considering software as being the weakest node in a software system (Allen, 2008), assuming it as a privileged target in security attacks since it could contain many forms of vulnerabilities, often not easily reparable to be fixed during development, exposing the whole hosting system to a wider range of potential risks.

DOI: 10.4018/978-1-4666-8473-7.ch024

Software vulnerabilities are often imprudently introduced because of the recruitment of unskilled developers and/or the adoption of inappropriate development processes which are unable to address specific requirements for secure software systems (Alkussayer, 2010; Goertzel, 2013). These vulnerabilities could be inserted at any step in a software development process (in specification, design and/or implementation) if underestimated, inadequately assessed and/or untreated.

Improving software security and safeguarding the information technology has been a long-term, complex and multifaceted problem as it requires multiple solutions and the application of many resources throughout the whole life cycle. Therefore, a considerable attention has been given to secure software engineering which is a new multidisciplinary domain that try to bridge the gap between software security and software engineering (Conklin & Dietrich, 2007; Hein, 2009; Siveroni, Zisman & Spanoudakis, 2010). Secure software engineering perspective is mainly interested in how to enhance existing lifecycle phases, artifacts and techniques used in each phase, or perhaps introduce new techniques, to support security. The holy grail of this field is software that is secure by construction. We believe that security will be improved only by focusing on its development process since the early phase (Hein, 2009).

Despite the multiple efforts undertaken in order to improve software security, recent statistics which were elaborated by ComScore (ComScore Inc, 2013), show that only 37% of Information Technology professionals cited that their organizations are building products and services with security in mind. Furthermore, 61% of developers are not currently taking advantage of built-in platform mitigation technologies that already exist.

This work tries to uphold secure software industry by supporting developers at all development stages and granularity levels. It contributes to the Secure Software Engineering domain by suggesting a multi-model process that formalizes and federates possible development processes which will be available through dynamic executions, and takes into account developers' affinities and software product states.

The present paper is structured as follows. First we briefly give a list of state-of-the-art of related works. This is followed by a presentation of our approach and its related concepts. Finally, to better explain its practice in a comprehensive way; a case study is given. The major contributions of this work and future works are emphasized in the conclusion.

2. RELATED WORKS

According to, McGraw's book (2006), "The software security field is a relatively new one. The first book and academic class on the topic appeared in 2001 (Viega & McGraw 2001), demonstrating how recent developers, architects and computer scientists have started systematically studying how to build secure software. The field's recent appearance is one reason why best practices are neither widely adopted nor obvious" (pp. 94).

Toward enhancing software security assurance, many research efforts tried to enrich software artifacts by adding some security aspects and/or to adopt new methodologies during some software development phases (Hussain, Rasool, Atef & Shahid, 2013). For example:

- Mc Dermott (1999) used misuse and abuse cases to capture and analyze harmful interactions that may occur between system and actors and to model systems' security threats in the requirements analysis phase,
- Jürjens (2002) extended UML notations (like UMLsec) to include modeling of security requirements in order to allow modeling access control mechanisms and aspects of information confidentiality,