Chapter 32
Building an Ambidextrous Software Security Initiative

Daniela Soares Cruzes
SINTEF Digital, Norway

Espen Agnalt Johansen
VISMA, Norway

ABSTRACT

Improving software security in software development teams is an enduring challenge for software companies. In this chapter, the authors present one strategy for addressing this pursuit of improvement. The approach is ambidextrous in the sense that it focuses on approaching software security activities both from a top-down and a bottom-up perspective, combining elements usually found separately in software security initiatives. The approach combines (1) top-down formal regulatory mechanisms deterring breaches of protocol and enacting penalties where they occur and (2) bottom-up capacity building and persuasive encouragement of adherence to guidance by professional self-determination, implementation, and improvement support (e.g., training, stimulating, interventions). The ambidextrous governance framework illustrates distinct, yet complementary, global and local roles: (1) ensuring the adoption and implementation of software security practices, (2) enabling and (3) empowering software development teams to adapt and add to overall mandates, and (4) embedding cultures of improvement.

INTRODUCTION

Today, nearly all sectors of society depend on software systems to operate efficiently. As the dependency on software has grown, so have the threats towards these systems and the potential consequences of incidents (Tøndel, Jaatun, Cruzes, & Moe, 2017). Though network security measures (such as firewalls and anti-virus software) can improve the security of the software systems, these only address the symptoms of the real problem: software that is crippled with vulnerabilities (McGraw, 2006).

DOI: 10.4018/978-1-6684-3702-5.ch032

Building security into the software through adopting software security activities and measures in the development process is a direct and effective way of dealing with cyber threats towards software systems (Tøndel, Jaatun, Cruzes, & Moe, 2017). This, however, adds to the development time and cost, and this addition needs to be well implemented to be effective. In many ways, security can be considered to be in conflict with the current trend of "continuous development" (Fitzgerald & Stol, 2017), reducing efficiency by delaying delivery of new features (at least in the shorter term, though costs may be saved through having to provide fewer fixes later).

Many researchers affirm that it may be more difficult to establish a working process for software security activities in agile development compared to waterfall-based development, where you could more easily have mandatory or recommended security activities for the different software development phases (Ben Othmane, Angin, Weffers, & Bhargava, 2014) (Ambler, 2008) (Microsoft, 2019). (Oyetoyan, Jaatun, & Cruzes, 2017) provide a brief overview of secure SDLs (Secure Development Lifecycle) and conclude that traditional approaches to software security do not necessarily work well with agile development processes. Additionally, security, as a non-functional requirement (NFR), is largely a systemic property, and with agile development it can be more of a challenge to have a complete view of the final system (Ben Othmane, Angin, Weffers, & Bhargava, 2014).

Non-functional requirements (NFRs) focus on aspects that typically involve or crosscut several functional requirements (Ambler, 2008). Although considered important and crucial to project success, it is common to see non-functional requirements losing attention in comparison to functional requirements. (Crispin & Gregory, 2009) argue that with that business partners might assume that the development team will take care of non-functional requirements such as performance, reliability, and security. But in reality, due to the agile philosophy that stimulates delivering user value early and often, the prioritization of quality attributes can be hard in early deliverable increments, resulting in hard-to-modify, unreliable, slow, or insecure systems (Baca, Boldt, Carlsson, & Jacobsson, 2015) (Bellomo, Gorton, & Kazman, 2015) (Wäyrynen, Bodén, & Boström, 2004). It is not rare to observe in software organizations that security practices are not prioritized, either because the practitioners are not able to see the relevance and importance of the activities to the improvement of the security in the project (Camacho, Marczak, & Cruzes, 2016) or because non-functional or cross-functional issues are perceived as a low risk for many systems (Jaatun, Cruzes, Bernsmed, Tøndel, & Røstad, 2015). Another issue is that agile development teams are generally composed of a small number of developers, and many times are composed of generalists. However, the proper handling of software security requires specialized tools and might need specialized knowledge. Given this need for specialized knowledge, a team member with specialized security skills might be required to avoid issues in production (Gregory & Crispin, 2014). As it is nowadays, there are usually not designated roles for security in the software development teams.

At the same time, agile development may come with some opportunities regarding security, e.g., to adapt to new security threats and to maintain the interaction with customers about security. (Tøndel, Jaatun, Cruzes, & Moe, 2017) propose a risk-centric approach to security. The authors found that the observed software security practices in software organizations were not based on an assessment of software security risks, but rather driven by compliance. Additionally, their practices could in many cases be characterized as arbitrary, late, and error driven, with limited follow up on any security issues throughout their software development projects. Based on the results of the study, the authors identified the need for improvements in three main areas: responsibilities and stakeholder cooperation, risk perception and competence, and practical ways of doing risk analysis in agile projects.