Chapter 1
Agile Software Development:
The Straight and Narrow Path to Secure Software?

Torstein Nicolaysen
NTNU, Norway

Richard Sassoon
NTNU, Norway

Maria B. Line
SINTEF ICT, Norway

Martin Gilje Jaatun
SINTEF ICT, Norway

ABSTRACT

In this article, the authors contrast the results of a series of interviews with agile software development organizations with a case study of a distributed agile development effort, focusing on how information security is taken care of in an agile context. The interviews indicate that small and medium-sized agile software development organizations do not use any particular methodology to achieve security goals, even when their software is web-facing and potential targets of attack. This case study confirms that even in cases where security is an articulated requirement, and where security design is fed as input to the implementation team, there is no guarantee that the end result meets the security objectives. The authors contend that security must be built as an intrinsic software property and emphasize the need for security awareness throughout the whole software development lifecycle. This paper suggests two extensions to agile methodologies that may contribute to ensuring focus on security during the complete lifecycle.

1. INTRODUCTION

A decade or so ago, the waterfall model was the favored way of managing/building projects, resulting in a very formal approach where security was handled both implicitly and specifically. Due to the rigid and formal nature of the waterfall model, there was a place for security in specific parts of the process. This does not automatically mean that the waterfall model will make the software secure; it still requires skilled people and determination to create secure software.

DOI: 10.4018/978-1-4666-1580-9.ch001

Agile software development has become a buzzword, and most modern IT-companies brag about how they are using it. Scrum (Scrum Alliance, 2009) is a popular and widely used agile software development methodology, which contains no specific techniques or help for handling critical elements like security. As Scrum is more of a project management methodology, it might not be up to Scrum to handle all aspects of security, but it does define how the requirements are elicited and how to communicate with the customer. If done by the book, the customer has to request security and then prioritize it. If neither the customer nor the developers are concerned with security, it will most likely never end up in the product backlog, and therefore it will be neglected.

This article refers to software security as the resistance against misuse and/or attacks. Specific security features such as login functionality and encrypted communication are part of this, but even more important is *secure code* features, aiming at making the code unexploitable, preventing attacks like buffer overflow, XSS and similar.

The big question is how software security fits into software development projects where agile methodologies are used. Can agile methodologies be mixed with the rigid and formal processes associated with software security, and if so, how?

This article presents an empirical study of how agile software developers include security in their projects. It also presents a case study showing that software development without a persistent focus on security results in software with a number of vulnerabilities. Finally, the article presents two possible extensions to agile methodologies, intended to increase developers' awareness of software security.

2. BACKGROUND

Enabling information systems to communicate via open networks such as the Internet will always be associated with elements of risk. (Mavridis, Georgiadis, Pangalos, & Khair, 2001) correctly state that "Security risks cannot be entirely removed when transmitting information over the Internet". The European Parliamentary Technology Assessment (EPTA) network has made similar considerations and specifically expressed concerns that privacy is challenged by the increase in development of ICT applications for the healthcare sector (EPTA, 2006). Such concerns are also raised by others, such as (Ilioudis & Pangalos, 2001) and (van der Haak et al., 2003).

(Boström, Wäyrynen, Bodén, Beznosov, & Kruchten, 2006) detail an extension to the XP planning game that is intended to establish a balance between the conventional (document-centric and plan-driven) way of doing security engineering, and the iteration-centric, feedback-driven XP practices. This is relevant as they try to solve a problem closely related to ours. The main difference is that they are specific to the XP methodology and only try to integrate the security requirements engineering (software security) activity, where as our approach is more generic for Agile methods and not focusing on just one specific security activity.

(Beznosov & Kruchten, 2004) attempt to find the pain points between agile methods and *security assurance*, and suggest some means on how to alleviate them. They group the problems and evaluate how good they match up against activities from security assurance. They focus on a specific problem, like Boström et al.'s approach, and do not seek to solve a more general problem.

(Siponen, Baskerville, & Kuivalainen, 2005) provide an example on how to integrate some security activities into agile development methods. They focus on four key security elements: security-relevant subjects, security-relevant objects, security classification of objects and subjects, and risk management. In the provided example where they apply their technique, it becomes apparent that it requires a lot more effort than what can be