Chapter 48
Creating and Applying Security Goal Indicator Trees in an Industrial Environment

Alessandra Bagnato
TXT e-solutions, Italy

Fabio Raiteri
TXT e-solutions, Italy

Christian Jung
Fraunhofer Institute for Experimental Software Engineering, Germany

Frank Elberzhager
Fraunhofer Institute for Experimental Software Engineering, Germany

ABSTRACT

Security inspections are increasingly important for bringing security-relevant aspects into software systems, particularly during the early stages of development. Nowadays, such inspections often do not focus specifically on security. With regard to security, the well-known and approved benefits of inspections are not exploited to their full potential. This book chapter focuses on the Security Goal Indicator Tree application for eliminating existing shortcomings, the training that led to their creation in an industrial project environment, their usage, and their reuse by a team in industry. SGITs are a new approach for modeling and checking security-relevant aspects throughout the entire software development lifecycle. This book chapter describes the modeling of such security goal based trees as part of requirements engineering using the GOAT tool dedicated plug-in and the retrieval of these models during the various phases of the software development lifecycle in a project by means of Software Vulnerability Repository Services (SVRS) created in the European project SHIELDS (SHIELDS - Detecting known security vulnerabilities from within design and development tools).

DOI: 10.4018/978-1-4666-4301-7.ch048

INTRODUCTION

Software security is still a challenging problem that affects software producers from small developer teams up to big vendors. The increasing complexity of software systems makes handling security ever more difficult. In addition, most software developers have insufficient knowledge regarding security aspects and use immature quality assurance techniques for preventing security defects across the entire software development lifecycle (SDLC), which aggravates the problem of secure software engineering.

Current quality assurance techniques for ensuring software security are, e.g., testing methods such as fuzzing (Sutton, Greene, & Amini, 2007), (Takanen, DeMott, & Miller, 2008), penetration testing (Arkin, Stender, & McGraw, 2005), or software inspections (Howard, 2006). Inspections – the systematic manual checking of a piece of software for certain defects – are one of the most effective and efficient quality assurance techniques (P. Runeson, C. Andersson, T. Thelin, A. Andrews, and T. Berling, 2006), (Wiegers, 2002). However, inspections often do not focus on security. Thus, the well-known and approved software inspections do not exploit their full potential regarding security. Adapting them to security needs is a challenging and time-consuming task, which requires appropriate security knowledge.

Detecting security-relevant defects too late in the development often leads to expensive corrections. Even worse is the deployment of faulty and insecure software, which may result in a bad reputation. Hence, interest in improving security inspections is widespread (Evans & Larochelle, 2002). In order to improve security inspections, a method for supporting inspections early in the SDLC has been developed. The approach provides structured reading support for the inspector during the inspection of a development artifact.

Security Goal Indicator Trees (SGITs), which are introduced in (Peine, Jawurek, & Mandel, 2008), were developed to improve guidance for focused security inspections. SGITs are tree-structured models in which the root node defines the general security goal. This goal is hierarchically decomposed into indicators that can be inspected independently. These indicators guide the inspector through the inspection process by subdividing complex security goals into a set of simple aspects that can be verified more easily. Best practices or principles of secure software engineering can be modeled as an SGIT. In order to achieve a specific security goal, the inspector has to map indicators of the model onto individual parts of the software or relevant parts from the specification documents in order to decide whether they have been fulfilled or not. Thus, our new inspection approach provides well-defined criteria that either have to be avoided in case the indicator might violate the achievement of the security goal (traditional inspection focus) or have to be fulfilled to reach the security goal (expanding the traditional inspection focus).

An inspector with little security background (e.g., a software developer) is able to perform security inspections by using this approach, due to the fact that the security knowledge is covered by the model (i.e., the SGIT). This reduces the burden on security experts and still permits to ensure security in software products. Thus, the approach using SGITs bridges the gap between security experts and software practitioners without any specific security expertise.

This book chapter discusses initial experience collected during the creation phase of new SGITs and describes the elicitation and modeling of new SGITs during the requirements analysis phase in an industrial environment. Furthermore, experiences gained from the process of deriving security goals and their indicators in the requirements engineering phase are discussed by TXT e-solutions S. p. A.

The book chapter outlines the approach using SGITs as reading support during inspection, the e-tourism project in which the technique was applied, the description of the software vulnerability repository service (SVRS) for storing the