The Journal of Systems and Software 139 (2018) 142–160

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A framework for semi-automated co-evolution of security knowledge and system models

Jens Bürger[a,*], Daniel Strüber[a], Stefan Gärtner[b], Thomas Ruhroth[a], Jan Jürjens[a,d], Kurt Schneider[c]

[a] University of Koblenz-Landau, Universitätsstraße 1, 56070, Koblenz, Germany
[b] adesso AG, Stockholmer Allee 200, 44269 Dortmund, Germany
[c] Lebniz University Hannover, Welfengarten 1, 30167 Hannover, Germany
[d] Fraunhofer ISST, Emil-Figge-Straße 91, 44227 Dortmund, Germany

ARTICLE INFO

Article history:
Received 1 April 2017
Revised 20 December 2017
Accepted 5 February 2018
Available online 6 February 2018

Keywords:
Security requirements
Software evolution
Co-evolution
Software design
Security impact analysis

ABSTRACT

Security is an important and challenging quality aspect of software-intensive systems, becoming even more demanding regarding long-living systems. Novel attacks and changing laws lead to security issues that did not necessarily rise from a flawed initial design, but also when the system fails to keep up with a changing environment. Thus, security requires maintenance throughout the operation phase. Ongoing adaptations in response to changed security knowledge are inevitable. A necessary prerequisite for such adaptations is a good understanding of the security-relevant parts of the system and the security knowledge.

We present a model-based framework for supporting the maintenance of security during the long-term evolution of a software system. It uses ontologies to manage the system-specific and the security knowledge. With model queries, graph transformation and differencing techniques, knowledge changes are analyzed and the system model is adapted. We introduce the novel concept of *Security Maintenance Rules* to couple the evolution of security knowledge with co-evolutions of the system model.

As evaluation, community knowledge about vulnerabilities is used (Common Weakness Enumeration database). We show the applicability of the framework to the *iTrust* system from the medical care domain and hence show the benefits of supporting co-evolution for maintaining secure systems.

1. Introduction

Security and privacy are critical success factors for software-intensive systems. Security flaws and data breaches may impair customer satisfaction and sales revenues, while entailing a high cost for flaw repair and compensations. Even if a system is built with great efforts to shield against the security[1] threats known at the time of its initial deployment, a challenging situation arises when the system is to be maintained for an extensive lifetime. Such *long-living systems* are particularly prone to security issues, since assumptions and design decisions made during their development may be invalidated when the environment changes: due to newly discovered attack types, increasingly ambitious security laws, and continuously evolving stakeholder requirements, employed security mechanisms may become obsolete.

Therefore, an achieved level of security must be *actively maintained* during the long-term evolution of a system.

For these reasons, it is crucial to support developers during the *detection* and *repair* of security flaws after the environment changes. It is desirable to enable automated support for these tasks as far as possible, relieving developers from unnecessary burden, while involving them whenever their input is necessary. From this goal, three main challenges arise. First, the automated detection of security flaws requires to leverage available knowledge on security threats. This knowledge needs to be managed explicitly and updated continuously, since it is subject to constant change. Second, to identify security flaws in a specific system when the context knowledge is updated, its security requirements need to be accounted for. To support the automated validation of these

* Corresponding author.
E-mail addresses: buerger@uni-koblenz.de (J. Bürger), strueber@uni-koblenz.de (D. Strüber), stefan.gaertner@adesso.de (S. Gärtner), ruhroth@uni-koblenz.de (T. Ruhroth), kurt.schneider@inf.uni-hannover.de (K. Schneider).
URL: http://jan.jurjens.de (J. Jürjens)

[1] In this paper, we consider privacy as a security aspect, acknowledging the ontological debate around these terms.

https://doi.org/10.1016/j.jss.2018.02.003
0164-1212/

requirements, they need to be maintained in a machine-readable form. Third, whenever violations of security requirements are detected, an immediate reaction becomes necessary: parts of the systems affected by vulnerabilities need to be identified; a suitable countermeasure needs to be determined and suggested to the security expert.

To address these challenges, in this paper, we present the SecVolution approach. The overall aim of SecVolution is to sustain the security of long-living systems whenever environmental changes have an impact on security properties. Our approach comprises three main components.

- *Security Context Knowledge* is expressed in terms of a *layered ontology* that allows the evolution of Security Context Knowledge to be managed and expressed in a formal manner. A central component of the layered ontology is an *upper ontology* of security-relevant concepts and their relationships.
- To detect threats in the system automatically whenever the Security Context Knowledge evolves, we enable the specification of *Essential Security Requirements* (ESRs). ESRs are amenable to an automated analysis against the Security Context Knowledge. To establish traceability between identified threats and the knowledge changes that provoked them, the analysis takes the *difference deltas* of the Security Context Knowledge as an input.
- Semi-automated reactions to co-evolve the security knowledge and the respective system models are specified using *Security Maintenance Rules*. As part of a Security Maintenance Rule, we employ model transformation rules to specify the system model evolution. A major benefit of this approach is that it can identify parts of the system models affected by security flaws automatically.

With this paper, we continue our ongoing work on the SecVolution approach (Ruhroth et al., 2014; Bürger et al., 2015), extending our earlier works in three main directions. First, the upper ontology presented here extends the one from our earlier work (Ruhroth et al., 2014) considerably by incorporating the results of a systematic literature review, allowing a more exact specification of security properties. Second, in our earlier work (Bürger et al., 2015), Essential Security Requirements were specified in natural language. With our new formalized notion of Essential Security Requirements, we provide a missing concept to enable an automated analysis of security requirements. Third, the aforementioned analysis, based on the formalized Essential Security Requirements and the Security Context Knowledge deltas, allows to detect potential threats in the system and map them to specific system artifacts.

To evaluate the extended SecVolution approach, we present a case study involving the open-source system iTrust. iTrust is a medical information system that fits well into our security setting.

As the evolution context, we use the privacy legislation of the European Union and Germany, comprising a set of privacy acts that have been changed repeatedly in the past years. Therefore, this setting is adequate to show the power of our approach in a realistic evolution scenario. The changes in the privacy legislation triggered changes in the underlying knowledge structure. The knowledge changes in turn called for changes to the system model of iTrust for recovering compliance to the privacy legislation. In sum, the evaluation shows the feasibility of our approach to identify security issues reliably and generate semi-automated reactions to security flaws.

The remainder of this paper is structured as follows: In Section 2, we sketch the SecVolution approach and define the scope of the research presented in this paper as well as relevant research questions. The evolution of environmental knowledge and a heuristic method to determine its impact on natural-language requirements is explained in Section 3.1. Based on this impact, the adaptation (or co-evolution) of the system model to retain a desired level of security is explained in Section 3.2 and Section 3.3. To evaluate our approach, we conducted a qualitative case study in Section 4 and discuss our results and insights. For this purpose, we used the medical care application *iTrust*. Related research in the field of security requirements and knowledge evolution as well as model co-evolution is listed in Section 5. In Section 6, we conclude our work and outline future research.

2. SecVolution approach and research challenges

According to Lehman and Ramil (2003), software evolution is the ongoing *progressive* change of software artifacts in one or more of their attributes over time. *Progressive* in this context means that the change results in improvement of the corresponding software. Each change preserves most properties (e.g. functionality and security) of the former system and is justified by a rationale. But changes may also lead to the emergence of new properties. Thus, evolution is caused by a wide variety of environmental changes such as technological changes, new stakeholders' needs, modified requirements and assumptions, changes in laws, rules as well as regulations, corrections of discovered problems and many others. Maintaining security of information systems by taking into account a continuously changing environment is therefore a challenging task in software engineering.

The term *co-evolution* is used in software engineering to describe the change of artifacts in response to a change in another artifact (cf. Mitleton-Kelly and Papaefthimiou, 2002). If artifact *A* evolves in response to changes in artifact *B*, *B* is called the causal artifact and *A* the effect artifact. Thus, co-evolution is the result of cause-effect changes between software artifacts. One reason for co-evolution is based on the fact that software artifacts depend on each other.

2.1. Overview of the SecVolution approach

The SecVolution approach is a holistic framework to deal with evolving knowledge in the environment of a software project. The overall goal is to restore security levels of an information system when changes in the environment put security at risk.

The SecVolution approach is the result of continuing research and extending the *SecReq* approach developed in our previous work (Houmb et al., 2009; Schneider et al., 2011; Jürjens and Schneider, 2014). As a core feature, SecReq supports reusing security engineering experience gained during the development of security-critical software and feeding it back into the model-based development process. To this end, SecReq combines three distinctive techniques to support security requirements elicitation as well as modeling and analysis of the corresponding system model: (1) Common Criteria (International Standardization Organization, 2007) and its underlying security requirements elicitation and refinement process, (2) the HeRA tool (Knauss et al., 2009) with its security-related heuristic rules, and (3) the UMLsec tool set (Jürjens, 2005) for secure system modeling and security analysis. This bridges the gap between security best practices and the lack of security experience among developers. However, a significant limitation of SecReq is that it cannot cope with evolution of the required security knowledge and, thus, has to be regarded as a "one–shot" security approach.

In SecVolution, to overcome this limitation of SecReq, the system's environment is monitored to infer appropriate adaptation operations. Fig. 1 depicts an overview of the resulting approach in the focus of this publication. Inputs to the approach are specification documents like (security) requirements, use cases, and misuse