International Journal of Secure Software Engineering
Volume 8 • Issue 3 • July-September 2017

Goal Modelling for Security Problem Matching and Pattern Enforcement

Yijun Yu, School of Computing and Communications, The Open University, Milton Keynes, UK
Haruhiko Kaiya, Kanagawa University, Hiratsuka, Japan
Nobukazu Yoshioka, GRACE Center, NII, Tokyo, Japan
Zhenjiang Hu, GRACE Center, NII, Tokyo, Japan
Hironori Washizaki, Waseda University, Tokyo, Japan
Yingfei Xiong, Peking University, Beijing, China
Amin Hosseinian-Far, Faculty of Business & Law, The University of Northampton, Northampton, UK

ABSTRACT

This article describes how earlier detection of security problems and the implementation of solutions would be a cost-effective approach for developing secure software systems. Developing, gathering and sharing similar repeatable programming knowledge and solutions has led to the introduction of Patterns in the 90's. The same concept has been adopted to realise reoccurring security knowledge and hence security patterns. Detecting a security problem using the patterns in requirements models may lead to its early prevention. In this article, the authors have provided an overview of security patterns in the past two decades, followed by a summary of i*/Tropos goal modelling framework. Section 2 outlines model-driven development, meta-models and model transformation, within the context of requirements engineering. They have summarised security access control types, and formally described role-based access control (RBAC) in particular as a pattern that may occur in the stakeholder requirements models. Then the authors used the i* modelling language and some elements from its constructs - model-driven queries and transformations - to describe the pattern enforcement. This is applied to a number of requirements models within the literature, and the pattern-based transformation tool they designed has automated the detection and resolution of this security pattern in several goal-oriented stakeholder requirements. Finally, the article also reflects on a variety of existing applications and future work.

KEYWORDS

Access Control, Goal Models, Model Transformations, RBAC, Security Patterns

1. INTRODUCTION

Today, security is considered as a key requirement in almost all systems. Implementation and architecting software systems with the incorporation of security considerations would be a cost-effective approach to preventive software maintenance. Security patterns were a programming rethinking approach so that security and safety features get embedded within the system at earlier stages of its development lifecycle. Patterns were initially proposed to outline recurring software engineering problems and solutions with a view to be used during software design phase (Vlissides, Helm, Johnson, & Gamma, 1995). It is also commonly recognised that there is a need for a pattern

DOI: 10.4018/IJSSE.2017070103

language to categorise security problems in software designs (Fernandez & Rouyi, 2001; Ruiz, Arjona, Mana, & Rudolph, 2017). Moreover, there have been further attempts to formalise and develop security patterns catalogues (Schumacher et al., 2013; Hamid, Gurgens, & Fuchs, 2016). These catalogues have gathered common solutions to known security problems using systems engineering and software development techniques. Identification and correction of errors early in the development lifecycle would cost less than fixing accumulated errors in design and implementation (Menzies, Nichols, Shull, & Layman, 2017). For security problems, therefore, one may ask a relevant question “Can we detect security problems and even resolve them early before it is too late?” This question has led to active research on the representation and analysis of security requirements (Liu, Yu, & Mylopoulos, 2003; Haley, Laney, Moffett, & Nuseibeh, 2008; Giorgini, Massacci, Mylopoulos, & Zannone, 2005; Souag, Mazo, Salinesi, & Comyn-Wattiau, 2016). There are various approaches for eliciting security requirements of a system-to-be. Abe, Hayashi, & Saeki (2015) propose an elicitation approach by which security functions are extracted from target documents, in line with the international standard of Common Criteria - ISO/IEC 15408. Riaz, Stallings, Singh, Slankas, & Williams (2016) have developed a framework within which security patterns are collected and security goals are identified. Work in the domain of this research assumes that security requirements can be elicited by arguing thoroughly about vulnerability in existing requirements models, such as trust assumptions (Haley, Robin, Moffett, & Nuseibeh, 2004), anti-goals (Van Lamsweerde, 2004; Li, Paja, Mylopoulos, Horkoff, & Beckers, 2016), misuse cases (Sindre & Opdahl, 2005; Ikram, Siddiqui, & Khan, 2014), abuse frames (Lin, Nuseibeh, Ince, Jackson, & Moffett, 2003), risk analysis (Massacci, Prest, & Zannone, 2005; Asnar et al., 2007; Souag et al., 2016), etc. Yet little has been done to suggest systematic changes in requirements models to resolve these vulnerabilities. Partly due to the fact that it is impossible to detect and resolve once-for-all the vulnerabilities, especially when not all problematic trust assumptions or anti-goals may be detected. Therefore, an appropriate approach would be helpful to slot in new security patterns. Capturing lessons learnt from the past as well-known security patterns, one may still need to enforce them to a requirements model for a new software project. In this paper, we have represented security patterns formally on basis of existing requirements modelling languages, such that an analysis tool can be developed to detect and resolve security problems in the modelled requirements. For such automated analysis, it is a reasonable assumption for stakeholder requirements to be modelled using a formal language: once requirements have been modelled as such, the tool could guarantee all instances of the security pattern can be detected and necessary changes can be suggested. To illustrate, we tried a single security pattern for recurring problems in role-based access control. Despite the possibility of selecting a different formal requirement modelling language for defining a security pattern, we selected elements from the goal oriented requirements modelling language i*/Tropos (Yu, 1996; Bresciani, Perini, Giorgini, Giunchiglia, & Mylopoulos, 2004) for two reasons. One, it is widely used in early requirements engineering which leads to many published models in the literature. Second, we have developed a tool support using the Eclipse Modelling Framework, which enables the techniques presented here. Our main contribution is to show that such formally defined security pattern can be directly used to detect and resolve the security problem on requirements models published in the literature. The key techniques used are model-driven query and transformation (Graaf, Weber, & Van Deursen, 2008) and model-driven development of requirements models (Budinsky, Brodsky, & Merks, 2003), both are integrated in our requirements engineering tools. The remainder of the paper is organised as follows. Section 2 explains key techniques used, including a language for modelling stakeholder requirements, model-driven software development and model-driven transformations; Section 3 briefly outlines access control mechanisms and types. The following section discusses the enforcing of a role-based access control pattern, and reviews several applications of the tool. Sections

Integrating DSLs into a Software Engineering Process: Application to Collaborative Construction of Telecom Services

Vanea Chiprianov, Yvon Kermarrec and Siegfried Rouvrais (2014). *Software Design and Development: Concepts, Methodologies, Tools, and Applications (pp. 570-595).*

www.igi-global.com/chapter/integrating-dsls-into-software-engineering/77723?camid=4v1a

Building Secure Software Using XP

Walid Al-Ahmad (2011). *International Journal of Secure Software Engineering (pp. 63-76).*

www.igi-global.com/article/building-secure-software-using/58508?camid=4v1a