Chapter 22
A Method for Model-Driven Information Flow Security

Fredrik Seehusen
SINTEF, Norway

Ketil Stølen
SINTEF, University of Oslo, Norway

ABSTRACT

We present a method for software development in which information flow security is taken into consideration from start to finish. Initially, the user of the method (i.e., a software developer) specifies the system architecture and selects a set of security requirements (in the form of secure information flow properties) that the system must adhere to. The user then specifies each component of the system architecture using UML inspired state machines, and refines/transforms these (abstract) state machines into concrete state machines. It is shown that if the abstract specification adheres to the security requirements, then so does the concrete one provided that certain conditions are satisfied.

INTRODUCTION

Security incidents occur on a daily basis within many companies. In CSI/FBI Computer Crime and Security Survey for 2005 (CSI/FBI, 2005), 74% of the companies reported security incidents. Despite the importance of security, careful engineering of security into overall design is often neglected and security features are typically built into an application in an ad-hoc manner or are only integrated during the final phases of system development (Lodderstedt et al., 2002).

DOI: 10.4018/978-1-4666-4301-7.ch022

Model-Driven Security (MDS) (Basin et al., 2006) advocates the opposite. MDS aims to raise the level of abstraction in design and development of secure systems by supporting

- A model-driven development process in which security is taken into account from start to finish,
- A clear separation of abstract, platform independent models (PIMs) and refined, platform specific models (PSMs), and
- Adherence preserving transformations between PIMs and PSMs.

Copyright © 2014, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

The central idea of MDS is that systems can be specified and shown to be in adherence with security requirements at different levels of abstraction. Abstraction is believed to simplify analysis, facilitate reuse of designs and early discovery of design flaws. At each level of abstraction, we distinguish between a system specification and a set of security requirements the system specification must adhere to. When we have established that a system specification adheres to the security requirements at a given level, this relationship should also hold at the next level so that the invested effort to establish adherence is not wasted. Hence, we would like adherence to be preserved under transformation. Adding details to a specification may of course require some additional analysis. However, it should not be necessary to recheck the adherence relationship already established at the more abstract level.

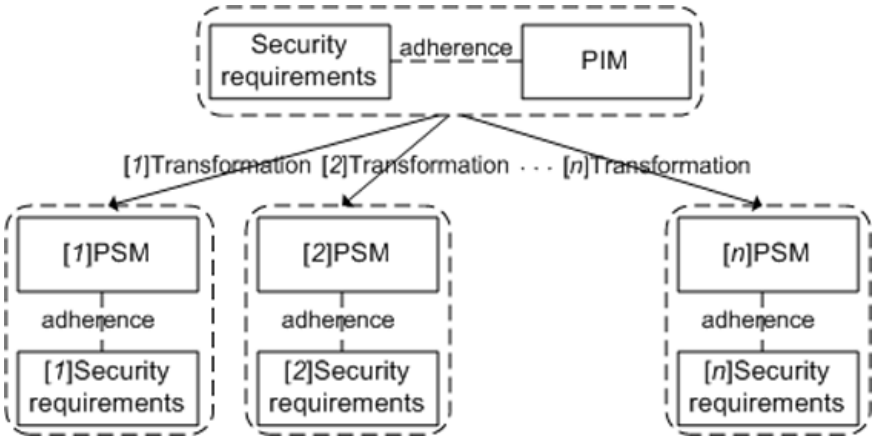
The MDS framework is illustrated in Figure1. Here a platform independent model together with its security requirements are transformed into several platform specific models with associated security requirements.

Already published approaches to MDS include (Basin et al., 2003, 2006; Breu et al., 2005; Burt et al., 2003; Fernández-Medina and Piattini, 2004; Heldal and Hultin, 2003; Lodderstedt et al., 2002; Vela et al., 2006). Although interesting, they are in most cases of a rather informal nature; the semantics of the languages used (at abstract or/ and concrete levels) are not sufficiently precise to allow for rigorous reasoning at more than one level of abstraction. Moreover, some of the approaches ((Basin et al., 2003, 2006; Burt et al., 2003; Lodderstedt et al., 2002)) consider transformation of security requirements only, and not transformation of system specifications. These approaches allow adherence checking only at the lowest level of abstraction. Others ((Breu et al., 2005; Burt et al., 2003; Fernández-Medina and Piattini, 2004; Vela et al., 2006)) do not clearly characterize what it means for a system to adhere to a security requirement. Instead, security is described in terms of a security mechanism.

Security is often defined as the preservation of confidentiality, integrity, and availability. In this chapter, we, however, focus on security in the more narrow sense of secure information flow properties (see e.g., (Bossi et al., 2004; Goguen and Meseguer, 1982; Mantel, 2000; McCullough, 1987; O'Halloran, 1990; Roscoe, 1995; Sutherland, 1986; Zakinthinos and Lee, 1997)) which provide an elegant way of specifying confidentiality as well as integrity requirements (Mantel, 2001).

The notion of transformation is closely related to refinement. That is, refinement is the (possibly

Figure 1. Model-driven security (pictured)



29 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/method-model-driven-information-flow/77717

Related Content

A Novel Spatial Data Pipeline for Streaming Smart City Data
Chase D. Carthen, Araam Zaremehrjardi, Vinh Dac Le, Carlos G. Cardillo, Scotty Strachan, Alireza Tavakkoli, Frederick C. Harris, Jr.and Sergiu M. Dascalu (2024). *International Journal of Software Innovation (pp. 1-15).*
www.irma-international.org/article/a-novel-spatial-data-pipeline-for-streaming-smart-city-data/359180

A Diffraction Service Composition Approach Based on S-ABCPC: An Improved ABC Algorithm
Xunyou Min, Xiaofei Xu, Zhongjie Wangand Zhizhong Liu (2022). *International Journal of Information System Modeling and Design (pp. 1-26).*
www.irma-international.org/article/a-diffraction-service-composition-approach-based-on-s-abcpc/300778

Toward a Statistical Characterization of Computer Daihinmin
Seiya Okubo, Yuta Kado, Yamato Takeuchi, Mitsuo Wakatsukiand Tetsuro Nishino (2019). *International Journal of Software Innovation (pp. 63-79).*
www.irma-international.org/article/toward-a-statistical-characterization-of-computer-daihinmin/217393

To Prevent Reverse-Enginnering Tools by Shuffling the Stack Status with Hook Mechanism
Kazumasa Fukudaand Haruaki Tamada (2015). *International Journal of Software Innovation (pp. 14-25).*
www.irma-international.org/article/to-prevent-reverse-enginnering-tools-by-shuffling-the-stack-status-with-hook-mechanism/126613

Collaborative Development of Dependable Cyber-Physical Systems by Co-Modeling and Co-Simulation
John Fitzgerald, Ken Pierceand Peter Gorm Larsen (2014). *Handbook of Research on Embedded Systems Design (pp. 1-28).*
www.irma-international.org/chapter/collaborative-development-of-dependable-cyber-physical-systems-by-co-modeling-and-co-simulation/116102