Preface

As software systems become more and more critical in every aspect of the human society, so does the demand to secure these systems. This is mainly because private information is stored in software systems and without enough security guarantees, organizations (and individuals) are not willing to share information or even use the technology. Even though security is an important issue, it is usually treated superficially and the usual security process is to add a standard set of security mechanisms, such as authentication, into the system.

However, it has been identified in many cases that securing software systems is not only about providing a set of standard security mechanisms. Fitting security enforcement mechanisms into an existing design, can lead to serious design challenges that usually translate into the emergence of computer systems afflicted with security vulnerabilities.

Providing adequate security requires the capability of reasoning about security. This means that software systems must be designed and deployed not only to meet certain functional requirements, but also to comply with the security requirements of the companies and/or organizations they are deployed at. In other words, security considerations must be integrated within software engineering practices to allow software system developers to consider security from the early stages of the development process.

Traditionally, software engineering deals with security as a non-functional requirement and usually considers it after the definition of the system. One of the reasons is that the research areas of software engineering and security engineering work independently. On one hand, software engineering techniques and methodologies do not consider security as an important issue, although they have integrated concepts such as reliability and performance, and they usually fail to provide precise enough semantics to support the analysis and design of security requirements and properties. On the other hand, security engineering research has mainly produced formal and theoretical methods, which are difficult to understand by non security experts and which, apart from security, they only consider limited aspects of the system.

This separation of work has resulted in an abstraction gap that makes the integration and practical application of security issues on modelling languages and software engineering methodologies difficult.

This book aims to provide the first step towards narrowing the gap between security and software engineering. To achieve this aim, this book (1) introduces the field of secure software engineering, a branch of research investigating the integration of security concerns into software engineering practices, which draws expertise from the security and the software engineering community; (2) introduces the problems and the challenges of considering security during the development of software systems; (3) it provides readers an understanding of the predominant theoretical and practical approaches that integrate security and software engineering by describing current secure software engineering approaches; and (4) it discusses future visions and directions for the field of secure software engineering.

ORGANIZATION OF THIS BOOK

This book is organized into three main sections: *Security Requirements Engineering*, *Modelling and Developing Secure Software Systems Using Patterns*, and *Modelling Languages and Methodologies for Secure Software Development*. The *Security Requirements Engineering* section (Section I) is organized into three chapters, Chapters II, III, and IV. The *Modelling and Developing Secure Software Systems Using Patterns* section (Section II) includes two chapters, Chapters V and VI. The *Modelling Languages and Methodologies for Secure Software Development* section (Section III) is organized into five chapters, Chapters VII, VIII, IX, X, and XI.

Additionally to these, we have the first chapter that introduces the problem of integrating security and software engineering, and a conclusive chapter that illustrates and explores challenges and future research directions of the field. A brief description of each of the chapters follows.

Chapter I (*Integrating Security and Software Engineering: An Introduction*) by H. Mouratidis and P. Giorgini, is an introduction to the current advances in the development of secure software systems. It provides an overview of the problem from the perspectives of security and software engineering, and introduces the field of Secure Software Engineering as a new branch of research concerned with the development of secure software systems, which integrates security and software engineering. Secure software engineering results in a situation where security is considered as part of the development process leading to the development of more secure software systems. In particular, the chapter discusses the research areas of software and security engineering are and it emphases the characteristics of these areas. Then the current state of the art on software and security engineering is presented, emphasizing the latest approaches to secure software engineering.

Chapter II (*Arguing Satisfaction of Security Requirements*) by C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh proposes an approach to carry out security requirements engineering, namely the process of eliciting, specifying, and analyzing

the security requirements for a system. The approach is founded on four main components. The first component is a framework that provides a systematic statement of the roles and relationships of security goals, security requirements, and security functions, and their relationships with other system and software requirements. The second is a way of describing threats and their interactions with the system. The third is a precise definition of security requirements. The fourth is a two-layer set of arguments to assist with validating the security requirements within the context of the system, to determine that the system is able to meet the security requirements placed upon it.

Chapter III (*Identifying Security Requirements Using the Security Quality Requirements Engineering (SQUARE) Method*) by N. R. Mead describes general issues in developing security requirements, methods that have been useful, and emphasize the system quality requirements engineering (SQUARE) method, which was developed by the CERT Program at Carnegie Mellon University's Software Engineering Institute, that can be used for eliciting, analyzing, and documenting security requirements for software systems. The method provides means for eliciting, categorizing, and prioritizing security requirements for information technology systems and applications. The SQUARE method seeks to build security concepts into the early stages of the development life cycle. The model may also be useful for documenting and analyzing the security aspects of fielded systems and could be used to steer future improvements and modifications to these systems.

Chapter IV (*A Social Ontology for Integrating Security and Software Engineering*) by E. Yu, L. Liu, and J. Mylopoulos describes the *i** agent-oriented modelling framework and how it can be used to treat security as an integral part of software system requirements engineering. The framework offers a set of security requirements analysis facilities to help users, administrators, and designers better understand the various threats and vulnerabilities they face, the countermeasures they can take, and how these can be combined to achieve the desired security results within the broader picture of system design and the business environment. The security analysis process is integrated into the main requirements process, so that security is taken into account from the earliest moment. The technology of smart cards and the environment surrounding its usage provides a good example to illustrate the social ontology of *i**.

Chapter V (*A Methodology to Develop Secure Systems Using Patterns*) by E. B. Fernandez, M. M. Larrondo-Petrie, T. Sorgente, and M. Vanhilst presents a methodology to build secure software for complex applications where patterns are used to help to apply security principles. The methodology considers the whole software lifecycle, uses security patterns, and is applied at all the architectural levels of the system. A main idea is that security principles should be applied at every stage and that each stage can be tested for compliance with security principles. The methodology shows how security patterns can be added to conceptual models in the analysis phase and how these analysis models are converted into design models with the addition of distribution and multiple architectural levels. Two running examples about a financial institution and a hospital are used to illustrate the different aspects of the proposed approach.

Chapter VI (*Modelling Security Patterns Using NFR Analysis*) by M. Weiss presents an approach where the main idea is to use non-functional requirements (NFR)

analysis to describe both the contributions patterns have on forces, and their design context. The level of structuring provided by NFR analysis can help to represent patterns in a more objective manner, and decide which patterns to apply in a given design context. The chapter describes how security requirements can be represented by this approach, but it also shows how the approach allows developers to consider security in the context of other non-functional requirements such as performance and scalability.

Chapter VII (*Extending Security in Agile Software Development Methods*) by M. Siponen, R. Baskerville, and T. Kuivalainen analyzes and outlines the requirements for security techniques to integrate seamlessly into agile methods. The analysis is presented through an example of an approach for adding security into agile information systems and software development methods. The chapter shows how this approach also offers a promising solution for adding security in agile information systems and software development, expanding earlier work that adapts it into the phases of agile methods.

Chapter VIII (*Modelling Security and Trust with Secure Tropos*) by P. Giorgini, H. Mouratidis, and N. Zannone describes how the integration of two prominent software engineering approaches, one that provides a security-oriented process and one that provides a trust management process results in the development of a methodology that considers security and trust issues as part of its development process. Such integration represents an advance over the current state of the art by providing the first effort to consider security and trust issues under a single software engineering methodology. Both approaches are extensions of Tropos, an- agent-oriented software development methodology. A case study from the health care domain is used to illustrate the result of the integration.

Chapter IX (*An Integrated Security Verification and Security Solution Design Trade-Off Analysis Approach*) by S. H. Houmb, G. Georg, J. Jürjens, and R. France describes a method that integrates security verification and security solution design trade-off analysis techniques. The security verification technique is used to verify that the solution has the required security level. The trade-off analysis technique is used to determine the best of the known solutions. The security requirements are precisely specified using UMLsec, an extension to UML for secure systems development. The security level of a solution can then be verified using UMLsec tool support. To evaluate the security solutions separately, the approach proposes to model them as security aspects using an aspect-oriented modelling (AOM) technique. These aspects are then evaluated and trade-off decisions are made based on computed Return on Security Investment (RoSI) for each security solution.

Chapter X (*Access Control Specification in UML*) by M. Koch, F. Parisi-Presicce, and K. Pauls discusses a methodology to integrate the specification of access control policies into UML. The methodology, along with the graph-based formal semantics for the UML access control specification, allows to reason about the coherence of the access control specification. The chapter also presents a procedure to modify policy rules to guarantee the satisfaction of constraints, and shows how to generate access control requirements from UML

diagrams. The main concepts in the UML access control specification are illustrated with an example access control model for distributed object systems.

Chapter XI (*Security Engineering for Ambient Intelligence: A Manifesto*) by A. Mana, C. Rudolph, G. Spanoudakis, V. Lotz, F. Massacci, M. Melideo, and J. S. Lopez-Cobo describes SERENITY, a comprehensive approach to overcome problems related to the design and engineering of secure and dependable systems for Ambient Intelligence applications. The key to success in this scenario is to capture security expertise in such a way that it can be supported by automated means. SERENITY's integral model of S&D considers both static and dynamic aspects. The combination of these innovations lays the foundations of an integrated, solid, flexible, and practical S&D framework for Ambient Intelligence ecosystems. The chapter aims at clarifying the challenges introduced in ambient intelligence ecosystems and pointing out directions for research in the different areas involved.

Finally, **Chapter XII** (*Integrating Security and Software Engineering: Future Vision and Challenges*) by H. Mouratidis and P. Giorgini concludes the book. The chapter lists and discusses nine challenges necessary for the advance of the secure software engineering field. The main idea behind each challenge is presented in a short sentence followed by a discussion which indicates why the challenge is important and in some cases the discussion provides ideas of how the challenge could be met.

Paolo Giorgini, Italy
Haralambos Mouratidis, UK
January 2006