Towards the Development of Secure Information Systems: Security Reference Diagrams and Security Attack Scenarios

H. Mouratidis[1], P. Giorgini[2] and G. Manson[3]

[1] School of Computing and Technology, University of East London, England
haris@uel.ac.uk
[2] Department of Information and Communication Technology, University of Trento, Italy
paolo.giorgini@dit.unit.it
[3] Department of Computer Science, University of Sheffield, England
g.manson@dcs.shef.ac.uk

Abstract. Security is one of the main challenges that developers of information systems face. However, current methodologies for information system development do not provide enough evidence of integrating successfully security concerns throughout the whole range of the development process. In this paper, we provide an approach towards the solution of two of the problems associated with the integration of security. Moreover, we describe how our approach can be integrated within the Tropos methodology and we illustrate it with the aid of a real-life example.

1 Introduction

One of the many challenges that developers face during the development of modern Information Systems (IS) is the security of the systems. Security Engineering of information systems is mainly concerned with methods providing cost effective and operationally effective protection of information systems from undesirable events [1], and as Anderson claims [2], security engineering is about building systems to remain dependable in the face of malice, error or mischance. Therefore, to effectively design a secure system it is important to know what the potential threats are, so that appropriate counter-measures can be taken.

However, no matter how good the protection is, possible attackers will (and have up to now) find possible vulnerabilities to expose the system. In addition, during the analysis and design developers, most of the time, assume the infrastructure is 100% trustworthy. However this might not be the case, making the prediction of every possible attack during the development of the system very difficult, and allowing a potential attacker to attack the system with types of attack that the developer cannot identify or foresee during the development of the system. On the other hand, a well-known axiom of computer security states that the only completely secure computer system is the one that has never been turned on.

Therefore, usually, the goal when developing an information system is to provide as much security as possible trading sometimes security requirements with other functional and non-functional requirements.

Although a methodology that provides solutions to all of the problems [3, 4, 5] that the integration of security in the development process can introduce, has not been developed, recently researchers have started to address some of these issues. McDermott and Fox adapt use cases [3] to capture and analyse security requirements, whereas Sindre and Opdahl [6] define the concept of a misuse case, the inverse of a use case, which describes a function that the system should not allow. In addition, a number of proposals [7, 8] that extend UML to cope with security have been proposed. In previous work [10, 11], we have proposed a structured process that integrates security and systems engineering, and it uses the same concepts and notations throughout the development of an information system, and we have integrated this process within the development stages of the Tropos methodology.

In this paper we extend our structured security process [10, 11] to provide solutions towards two important problems: the involvement of novice security developers in the development of information systems that usually require knowledge of security, and the difficulty of fully testing a proposed security solution at the design stage. In particular, we introduce the security reference diagram[1] and the security attack scenarios. To demonstrate their applicability we use as an example the electronic single assessment process system case study and we explain how the proposed solutions can be integrated into the Tropos security process.

Section 2 of the paper introduces the case study that will be used throughout the paper to illustrate the proposed approach, and Section 3 introduces the security reference diagram. Section 4 describes the Security Attack Scenarios and Section 5 concludes the paper and it provides some directions for future work.

2 The Case Study

In this section, we present the electronic Single Assessment Process (eSAP) system case study, first introduced [17] by one of the authors, which we will use throughout this paper to illustrate our approach.

The eSAP case study involves the development of a health and social care information system for the effective care of older people in England [17]. Security is a very important factor in the development of the electronic Single Assessment Process, since security of personal health information is considered priority by many health care unions in different countries of the world including England. The privacy of health and social care information, such as the health and social care plans used in the electronic single assessment process, is the number one security concern when developing the system.

[1] Although we have introduced the term in previous papers [10, 11], this is the first paper that the security reference diagram is explicitly described and it is integrated within the development process

Other important concerns are integrity and availability. Integrity assures that information is not corrupted and availability ensures the information is always available to authorised health and social care professionals.

3 Security Reference Diagram

Tropos [12] is a software development methodology that employs modelling concepts such as actors, goals, soft goals, tasks, resources and intentional dependencies. These concepts are suited to model security requirements, since these (security requirements) are usually expressed in natural language using notions such as agents and high level goals such as confidentiality and authentication [9].

In previous work [10, 11], we have introduced extra concepts to the Tropos methodology and we have also redefined existing concepts with security in mind, to enable Tropos to model security requirements during the software development process. However, the identification of security requirements in the very early phase of the software development is still a problematic activity. In order to help developer in this activity we introduce the security reference diagram.

The *security reference diagram* represents the relationships between security features, threats, protection objectives, and security mechanisms and its main purpose is to provide security novices with a valuable reference point when considering security issues during the development of information systems.

Security features represent features associated to security that the system-to-be must have. In this work the concept of a soft-goal is used to capture security features on the security reference diagram because security features are not subject to any clear criteria for satisfaction. Examples of security features are privacy, availability, and integrity. *Protection objectives* represent a set of principles or rules that contribute towards the achievement of the security features. In this work, protection objectives are modelled using the concept of a goal, because a protection objective represents desired security states that the system must have. Examples of protection objectives are authorisation, cryptography and accountability. *Security mechanisms* represent standard security methods for helping towards the satisfaction of the protection objectives. The concept of a task is used to model security mechanisms because a security mechanism represents a particular way of satisfying a protection objective. It must be noticed that furthered analysis of some security mechanisms is required to allow developers to identify possible security sub-mechanisms. *Threats* represent circumstances that have the potential to cause loss or problems that can endanger the security features of the system. Since Tropos notation does not provide any related concept to model threats, a new notation has been introduced. Examples of threats are social engineering, password sniffing and eavesdropping attacks. The notation of these concepts is illustrated in Figure 1. The above-mentioned nodes of a security reference diagram are associated with the aid of two types of links (similar to the contribution links that can be found in the Tropos methodology): *positive* and *negative* contribution links. A positive contribution link associates two nodes when one node helps in the fulfilment of the other. Consider, for instance, a protection objective that contributes positively to the satisfaction of a security feature. A negative contribution link, on the other hand, indicates that a node contributes towards the denial of another node. Graphically, a positive contribution link is modelled as an arrow, which points towards the node that is satisfied, with a plus (+) whereas a negative contribution link is represented as an arrow with a minus (-) (see Figure 2 as an example).

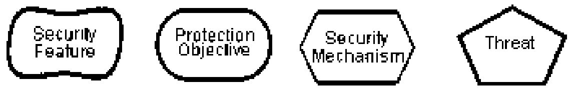

Fig. 1. Notation used in the security reference diagram

The process of constructing the security reference diagram is non-deterministic due to the choice of a particular rule, at each step. To control this kind of non-determinism during the construction of the security reference diagram, priority rules have been assigned. These rules, introduced in [16], are presented below in priority sequence: **R1**. Introduce the security feature to the diagram, **R2**. Introduce the security threats and associate them with the security features, **R3**. Introduce the protection objectives and associate them with the security features, **R4**. Introduce the security mechanisms and associate them with the protection objectives, **R5**. Decompose the security mechanisms to security sub-mechanisms.

Consider for instance, the construction of the security diagram for the eSAP system. As identified in Section 2, the main security features of the system are *privacy*, *integrity* and *availability* as shown in the security reference diagram in Figure 2. As shown in the diagram, each of those security features receives (according to the second rule (R.2)) some negative contributions from different security threats. For example, social engineering is a major threat to the system. This involves a private detective (or someone interested in obtaining personal health information) that calls in the health professional's office, introduces himself as a doctor in an emergency or acute hospital and asks information about the medical record of a particular patient [2].

Furthermore, the size of the electronic single assessment process system and the large number of health and social care professionals that might be involved introduces the problem of data aggregation and increases the risk of social engineering or unauthorised access. Apart from the threats to the privacy of the data, there are threats to the integrity and the availability of it. For instance, malicious attackers might change the content of medical care plans, or a number of compromised systems attack a single target. This initially results in denial of service to the users of the targeted system, and later in the shut down of the system, therefore making the system unavailable. On the other hand, the security features receive (according to the third rule (R.3)) some positive contributions from different protection objectives. For instance, Privacy receives positive contributions from Authorisation and Cryptography. In turn, each of these protection objectives receives some positive contributions (according to the fourth rule (R.4)) from different security mechanisms. For example, the Authorisation protection objective receives positive contributions from the Access Control, Authentication and Information Flow security mechanisms. Moreover, these security mechanisms can be decomposed (according to the fifth rule

(R.5)) to sub-mechanisms. For instance, the Authentication security mechanism is decomposed to Passwords, Digital Signatures and Biometrics.

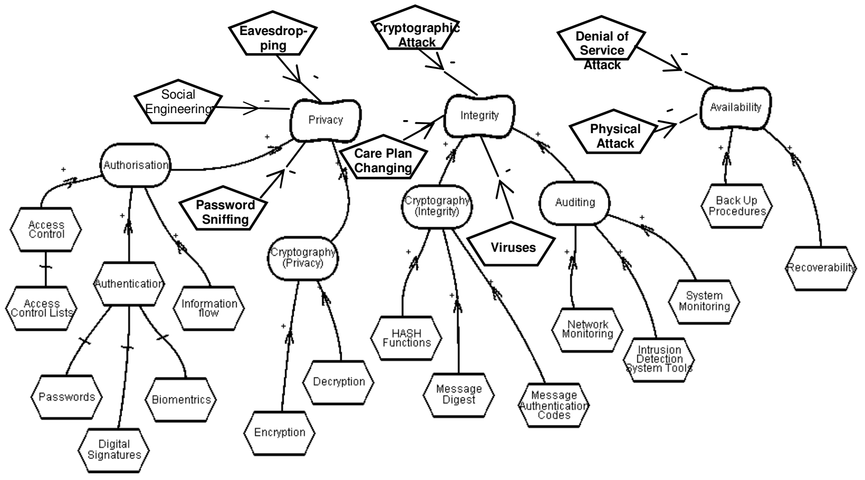

Fig. 2. An example of a security reference diagram

As presented in [10, 11], during the rest of the early requirements analysis, the late requirements analysis and the architectural design, the actors of the eSAP system along with their secure capabilities and the interactions between them are identified. As a result of this analysis [16], the eSAP actor is decomposed, amongst others, to the following actors: Skills Manager, Assessment Evaluator, Integrity Verification Manager, Professional DB Manager, Authenticator, Actions Manager, Auditing Manager, Access Controler, and eSAP guard.

Having identified the actors of the system, the next step (during the architectural design phase) involves the application of the Security Attack Scenarios to allow developers to test the security of the developed system against possible security attacks.

4 Security Attack Scenarios

Testing is widely considered an important activity that helps to identify errors in a system and techniques such as control and data flow testing, formal specifications, special testing languages, and test tools have been used for many years, in testing systems, and they are considered valuable solutions for many projects. However, most of these approaches are difficult to apply, they require special training and skills, and they employ their own concepts and notations [13]. As a result, the applicability of those approaches conflict with some of the requirements [16] which a security-oriented approach should demonstrate. Therefore, a technique, which is based on the use of scenarios has been developed and integrated within the Tropos security-oriented process to enable developers to test the system under development.

A Security Attack Scenario (SAS) is represented as enhanced Tropos diagrams and is used to test how the system copes in different kinds of security attacks. A Security Attack Scenario involves a possible attacker, possible attack(s), the resources that are attacked, and the actors of the system related to the attack together with their secure capabilities. An attacker is depicted as an actor who aims to break the security of the system. The attacker intentions are modelled as goals and tasks and their analysis follows the same reasoning techniques that the Tropos methodology employs for goal and task analysis. Attacks are depicted as dash-lined links (called attack links) that contain an “attacks” tag, starting from one of the attackers goals and ending to the attacked resource (see for example Figure 3).

The process is divided into three main stages: creation of the scenario, validation of the scenario, and testing and redefinition of the system according to the scenario. There are two basic steps in the creation of a scenario. The first step involves the identification of the attackers’ intentions and the possible attacks to the system and the second step involves identification of possible countermeasures of the system to the indicated attacks.

During the first step, Tropos goal diagram notation is used for analysing the intentions of an attacker in terms of goals and tasks. Some of these goals can be identified by the threats modelled on the security reference diagram. For example, the threat “social engineering” can introduce a goal “perform social engineering” to a potential attacker. However, other goals (apart from the ones introduced by the threats identified in the security reference diagram) could be derived from the analysis of a possible attacker’s intentions. When the analysis of the attacker’s intentions has been completed, possible attacks to the resources of the system are indicated using attack links.

The next step in the creation of a security attack scenario involves the identification of the actors of the system that possess capabilities to prevent the identified, from the previous step, attacks. Therefore, the actors of the system related to the identified attack(s) are modelled. The secure capabilities, of each actor, that help to prevent the identified attacks are identified and dashed-links (with the tag “help”) are provided indicating the capability and the attack they help to prevent.

When the scenarios have been created, they must be validated. Therefore, the next stage of the process involves the validation of the scenario. Software inspections are proved as an effective means for document-based validation [15] and as such are the choice of this research for the validation of the security attack scenarios. The inspection of the scenarios involves the identification of any possible violations of the Tropos syntax and of any possible inconsistency between the scenarios and the models of the previous stages. Such an inspection involves the use of validation checklists.

When the scenarios have been validated, the next step aims to identify test cases and test, using those test cases, the security of the system against any potential attacks. Each test case is derived from a possible attack depicted in the security attack scenarios. For each test case a *precondition* is necessary (the state of the system before the attack), an *expected system reaction* (how the system reacts in the attack), and also a *discussion* that forms the basis for the decision regarding the test case.

The test cases are applied and a decision is formed to whether the system can prevent the identified attacks or not. The decision whether an attack can be prevented (and in what degree) or not lies on the developer. However as an indication of the decision it must be taken into consideration that at least one secure capability must help an attack, in order for the developer to decide the attack can be prevented. Attacks that cannot be prevented are notated as solid attack links (as opposed to dashed attack links). For each attack that it has been decided it cannot be prevented, extra capabilities must be assigned to the system to help towards the prevention of that attack

To better understand the process, consider the eSAP case study. As it can been seen from the security reference diagram, three are the main security features required by the eSAP system: privacy, integrity and availability. Different categories of attacks can be identified [18] that can endanger these security features.

Due to lack of space, in this paper we only present a security attack scenario related to a modification attack [18]. The modification scenario involves an ***Attacker*** that wishes to attack the integrity of the ***eSAP*** system. As identified in the analysis of the security reference diagram, three main threats are involved in this kind of attack, cryptographic attacks, care plan changing and viruses.

Therefore, the ***Attacker***'s main goal, ***attack eSAP integrity***, can be decomposed to modify content of messages, change values in data files, and alter programs to perform differently (Figure 3). The first sub-goal involves the ***Attacker*** trying to modify the content of any messages transmitted over the network. To fulfil this goal, the ***Attacker*** might try to employ cryptographic attacks to any resource transmitted between any external actors and the ***eSAP*** system. The second sub-goal indicates the Attacker trying to change the values in data files of the system. The fulfilment of this goal can be satisfied by means of changing the data of resources stored in the ***eSAP*** system. The third sub-goal indicates the attempt of the ***Attacker*** to alter a program so it performs differently. Mainly this can be achieved using viruses that can alter the behaviour of specific programs in order to enable the attacker to gain access to the system or to system's information.

As an example, consider the scenario in which a ***Social Worker*** wishes to obtain an assessment evaluation [16]. Three main test cases are identified, cryptographic attacks, data changing attacks and viruses attacks. Due to lack of space, only the viruses attack test case is shown in figure 4.

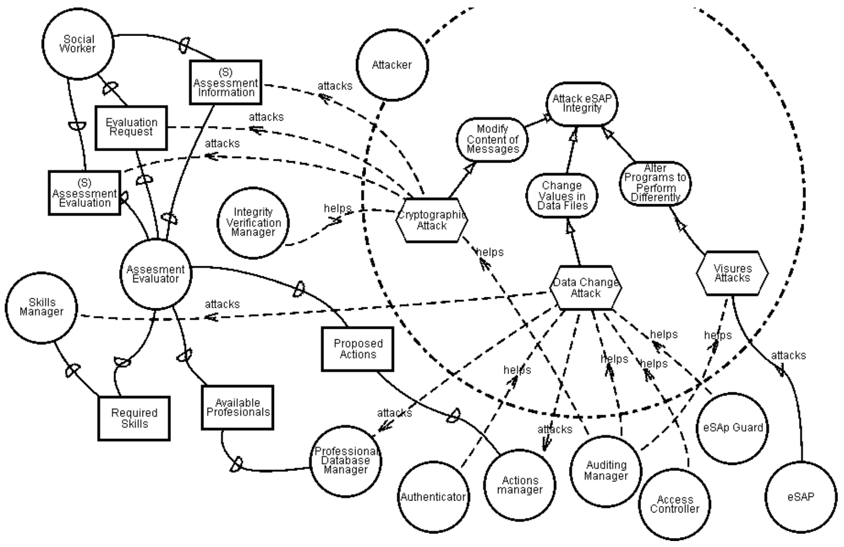

Fig. 3. Modification Attack Scenario

Test Case 3: Viruses
Precondition: The ***Attacker*** tries to change the system behaviour by using some kind of virus.
System expected security reaction: The system should be able to prevent viruses.
Discussion: Viruses consist one of the most sophisticated threats to computer systems. It is quite common for attackers to send viruses to computer systems they want to attack in order to exploit vulnerabilities and change the behaviour of the system. Although many effective countermeasures have been developed for existing types of viruses, many new types of viruses are also developed frequently. An ideal measurement against viruses is prevention. In other words, viruses should not get into the system. However, this is almost impossible to achieve. Therefore, the best approach is to be able to detect, identify and remove a virus. Auditing helps towards the detection of the virus. However, apart from this the eSAP system is not protected against viruses.
Test Case Results: The ***eSAP*** system needs to be integrated with an anti-virus program to enable it to effectively detect, identify and remove any possible viruses. Such a program, which could be another internal actor of the ***eSAP*** system, should be able to monitor the system and take effective measurements against any possible viruses.

Fig. 4. An example of a test case

7 Conclusions and Future Work

The introduction of the security reference diagram allows the identification of desired security requirements very early in the development stages, and helps to propagate them throughout the development stages. This introduces a security-oriented paradigm to the software engineering process. This and the fact that security expert developers can expand the security reference diagram, provides security novices with a valuable reference point when considering security issues during the development of information systems.

It is worth mentioning that the security reference diagram is similar to the security catalogue first introduced by Yu [19]. The main difference lies in the concepts introduced by the security reference diagram and also on its integration within the development stage of the Tropos methodology.

On the other hand, the introduction of security attack scenarios to test the system's response to potential attacks allows developers to test the developed security solution during the early development stages. The proposed scenario-based analysis is similar to the work presented by Liu et al [20]. However, there are some important differences. Liu's work is basically used to identify security requirements; the Security Attack Scenarios in this work are used to test the security requirements of the system identified in the previous development stages. So a very similar idea is applied in a different stage of the development lifecycle. Liu argues that when the intentions of the attackers are identified the system can be equipped with countermeasures, however it is never mentioned how this can be done neither provides a kind of process for providing such countermeasures. Moreover, in the approach of this research test cases are considered. In other words, a way is provided to test each scenario for specific test cases, reason about the reaction of the system and take a final decision if the system can react to the specific attack. In cases that the system cannot react to the attack, possible countermeasures are discussed and secure capabilities are introduced to the actors of the system to satisfy them.

Future work involves the expansion of our approach to consider security related concepts such as trust, and ownership and also the development of a tool to assist in the development process by, for example, automatically produce some attack scenarios and check the system responses.

References

1. V. P. Lane, "Security of computer based information systems", Macmillan Education Ltd, 1985
2. R. Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems", Wiley Computer Publishing, 2001
3. J. McDermott, C. Fox, "Using Abuse Care Models for Security Requirements Analysis", Proceedings of the 15th Annual Computer Security Applications Conference, December 1999.
4. H. Mouratidis, P. Giorgini, M. Schumacher, G. Manson. *Security Patterns for Agent Systems*. Proceedings of the Eight European Conference on Pattern Languages of Programs (EuroPLoP), Irsee, Germany, June 2003.
5. L. Chung, B. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach", Proceedings of the 17th International Conference on Software Engineering, Seattle- USA, 1995
6. G. Sindre, A. L. Opdahl, "Eliciting Security Requirements by Misuse Cases", Proceedings of TOOLS Pacific 2000, November 2000.

7. T. Lodderstedt, D. Basin, J. Doser, “SecureUML: A UML-Based Modelling Language for Model-Driven Security”, in the Proceedings of the 5th International Conference on the Unified Modeling Language, 2002.
8. Jan Jürjens, “Towards Secure Systems Development with UMLsec”, Fundamental Approaches to Software Engineering (FASE/ETAPS) 2001, International Conference, Genoa 4-6 April 2001
9. P. Giorgini, F. Massacci, J. Mylopoulos, “Requirement Engineering meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard”, in Proceedings of the 22nd International Conference on Conceptual Modeling, Springer, 2003
10. H. Mouratidis, P. Giorgini, G. Manson, “Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems” Lecture Notes in Computer Science 2681, page 63-78, ISBN 3-540-40442-2, Springer 2003
11. H. Mouratidis, P. Giorgini, G. Manson, “Modelling secure multiagent systems”, in the proceedings of the Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, pages 859-866, ISBN 1-58113-683-8, ACM 2003.
12. A. Perini, P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos. “Towards an Agent Oriented Approach to Software Engineering. In A. Omicini and M.Viroli, editors, WOA 2001 – Dagli oggetti agli agenti: tendenze evolutive dei sistemi software, Modena-Italy, September 2001.
13. J. Ryser, M. Glinz, “A Practical Approach to Validating and Testing Software Systems Using Scenarios”, Proceedings of the Third International Software Quality Week Europe (QWE'99), Brussels, Belgium, November, 1999.
14. B. Schneir, “Secrets and Lies: Digital Security in a Networked World”, John Willey and Sons, 2000.
15. G. Kosters, B. U. Pagel, M. Winter, “Coupling Use Cases and Class Models”, Proceedings of the BCS-FACS/EROS Workshop on “Making Object Oriented Methods More Rigorous”, Imperial College, London-England.
16. H. Mouratidis, “Extending the Tropos Methodology to Accommodate Security Issues”, Internal Report, University of Sheffield, August 2003.
17. H. Mouratidis, I. Philp, G. Manson, “Analysis and Design of eSAP: An Integrated Health and Social Care Information System”, in the Proceedings of the 7th International Symposium on Health Information Managements Research (ISHIMR2002), Sheffield, June 2002
18. W. Stallings, “Cryptography and Network Security: Principles and Practice”, Second Edition, Prentice-Hall 1999.
19. E. Yu, L. Cysneiros, “Designing for Privacy and Other Competing Requirements”, in Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS’02). Raleigh, North Carolina, October 2002.
20. Liu, E. Yu, J. Mylopoulos, “Analysing Security Requirements as Relationships Among Strategic Actors”, 2nd Symposium on Requirements Engineering for Information Security (SREIS’02). Raleigh, North Carolina, October 2002.