Analysing Security in Information Systems

Haralambos Mouratidis[1], Paolo Giorgini[2]

[1] School of Computing and Technology, University of East London, England
h.mouratidis@uel.ac.uk
[2] Department of Information and Communication Technology, University of Trento, Italy
paolo.giorgini@dit.unitn.it

Abstract. The last few years there is a tendency to integrate security issues during the development of information systems with the aim to develop more secure information systems. In this paper we propose an analysis, based on the measures of criticality (how critical a component of the system is) and complexity (represents the effort required by the components of the system to achieve the requirements that have been imposed to them), which aims to identify possible bottlenecks of an information system with respect to security. An integrated health and social care information system is used as a case study throughout this paper.

1 Introduction

It is widely accepted that security of information systems is an important concern. However, security is, usually, not considered during the development stages and it is added as an afterthought [Dev00]. This introduces problems to the system development. Therefore, methodologies must provide a systematic approach to assist developers when dealing with security issues.

In this paper we propose an approach that identifies the impact of each component to the security of the system and the effort that is required by each system component to achieve the security requirements that have been imposed to them.

Section 2 provides an overview of the Tropos methodology and it describes how the methodology handles security. Section 3 introduces an example that is used in the rest of the paper to illustrate our approach, whereas in Section 4 the concepts of security criticality and security complexity are defined and the process of analysing the complexity and criticality of an information system with respect to security is described. Finally, Section 5 presents some concluding remarks and directions for future work.

2 Tropos

Tropos [Bre02] is a requirements driven development methodology that uses concepts such as actors, goals, tasks, and social dependencies adopted by the i* framework [Yu95].

A dependency between two actors represents that one actor depends on the other to attain some goal, execute a task, or deliver a resource. The depending actor is called the depender and the actor who is depended upon is called the dependee. The type of the dependency describes the nature of an agreement (called dependum) between dependee and depender.

Tropos spans across five stages of the software development process: Early Requirements Analysis, Late Requirements Analysis, Architectural Design, Detailed Design and Implementation. It is worth mentioning that in this paper we integrate our approach only in the early requirements analysis stage of the Tropos methodology.

The security process in Tropos is one of analysing the security needs of the stakeholders and the system in terms of security constraints [Mou03] imposed to the stakeholders (early requirements) and the system (late requirements), identifying secure entities [Mou03] that guarantee the satisfaction of the security constraints, and assigning capabilities to the system (architectural design) to help towards the satisfaction of the secure entities. Additionally to security constraints, Tropos defines the term secure entity to describe any goals and tasks related to the security of the system. A secure goal [Mou03b] represents the strategic interests of an actor with respect to security. The precise definition of how the secure goal can be achieved is given by a secure task. A secure task [Mou03b] is defined as a task that represents a particular way for satisfying a secure goal. A secure dependency [Mou03b] introduces security constraint(s) that must be fulfilled for the dependency to be satisfied. The notation for the above presented concepts is shown in figure 1.

3 An example

We use throughout the paper a case study from the health and social care setting; the electronic Single Assessment Process system (eSAP). The eSAP system is an information system for the effective care of older people, in which security is considered priority [Mou03c]. During the early requirements analysis stage, the dependencies, the goals and the security constraints between the actors of the eSAP can be modelled using Tropos actors' diagram as shown in figure 1. In such a diagram each node represents an actor, and the links between the different actors indicate that one depends on another to accomplish some goals.

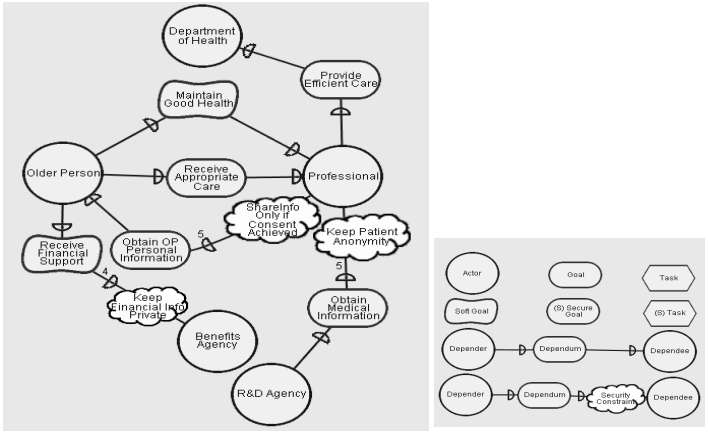

Figure 1*:* The actor diagram of the eSAP case study.

For instance, in our example, the *Older Person* depends on the *Benefits Agency* to *Receive Financial Support*. However, the *Older Person* worries about the privacy of their finances so they impose a security constraint to the *Benefits Agency* actor, to keep their financial information private.

4 Security Criticality and Security complexity

A careful examination of the actor diagram (Figure 1) indicates that different actors influence the security of the eSAP system differently (depending on how many security constraints they have been imposed to).

For example, the Professional actor has been imposed two security constraints and as a result she would influence the security of the eSAP system more than the Department of Health actor who has not been imposed any security constraints.

Therefore, it is important to provide an analysis that identifies the impact each actor has on the security of the system. This will help us to identify possible security bottlenecks of the system, and refine the system in order to avoid them. Moreover, we need to evaluate how much effort is required by each actor to achieve the security constraints that have been imposed to them. To help us with this analysis we introduce the measures of security criticality and security complexity and we define them as follows:

Security Criticality is the measure of how the security of the system will be affected if the security constraint is not achieved, whereas, **Security Complexity** is the measure of the effort required by the responsible actor for achieving a security constraint.

To allow the more precise calculation of security criticality, we define ingoing criticality and outgoing criticality. Ingoing security criticality is the security criticality that actors assume when they are responsible for achieving a security constraint. Outgoing security criticality represents the security criticality of the achievement of a constraint for the imposer (the actor who imposed the security constraint).

To calculate the criticality of the system, the dependencies between the different actors must be considered and a value should be assigned for each security constraint. Moreover, a maximum value of criticality should also be defined, for each actor, taking into account the actor's abilities, available time, and the responsibilities they have in the organization.

In our analysis we have assumed that criticality obtains integer values within the range 1-5, where 1 = very low, 2 = low, 3 = medium, 4 = high, 5 = very high.

It worth mentioning that in the case of an open secure dependency (a dependency that has no security constraints attached to it), security criticality (both ingoing and outgoing) assumes a value of zero.

On the presented example, we have assigned values for each security constraint (next to security constraints in Figure 1) after closely studying the system's environment and discussing it with it the stakeholders. For instance, we have assigned 5 for the "share info only if consent achieved" and "Keep patient anonymity" security constraints imposed by the *Older Person.*

As a result, the actors assume the criticality values shown in the following table.

Table 1: Criticality values of the actors of eSAP

Actor	Ingoing Criticality	Outgoing Criticality
Older Person	0	9 (5+4)
Benefits Ag.	4	0
Professional	10 (5+5)	0
DoH	0	5
R&D Agency	0	0

On the other hand, taking into account the security complexity helps to design subsystems to support actors that might be in danger of not achieving some security constraints that have been imposed and as a result endanger the overall security of the system.

In order to be realistic, we need to take into account both the system and security complexity, where *System Complexity* is defined as the measure of the effort required from the dependee to achieve the dependum [Gar02].

Similar to criticality analysis, we have assumed, for reasons of simplicity, that complexity (system and security) can obtain integer values within the range 1-5. Also similarly to criticality, a maximum value of (overall) complexity is defined for each actor. Moreover, in cases where the dependum is a soft goal, minimum system complexity values are assumed. To be able to precisely assign values for security and system complexity, each actor of the system and their security constraints and goals respectively must be further analysed. For this analysis, we are employing Tropos rationale diagrams [Bre02]. Each rationale diagram analyses the internal goals, security constraints and dependencies of each actor (figure 2).

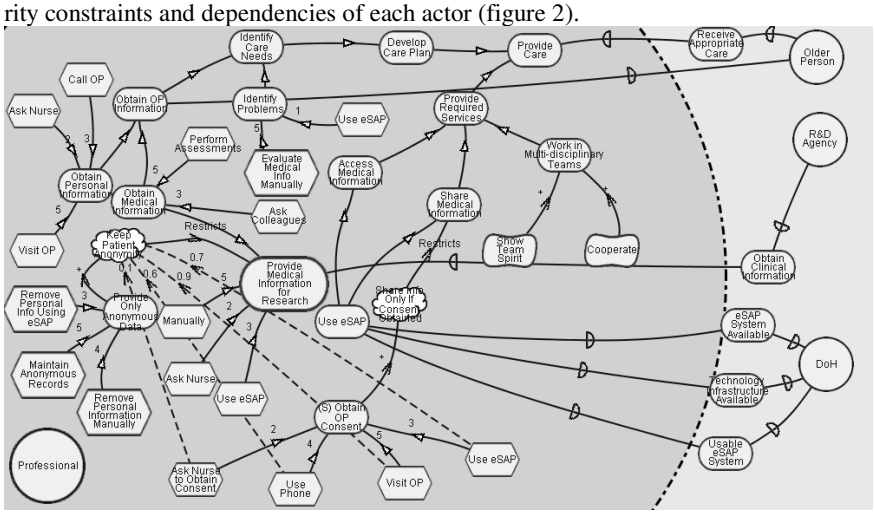

Figure 2: Professional actor goal diagram

The first step in calculating the complexity is to assign weights (with respect to the system complexity) to the different relationships involved in the satisfaction of the security constraints (secure goals), that have been imposed to the actor, and the actor's strategic goals.

Consider for instance, the rationale diagram of the *Professional* actor (shown in figure 2). To satisfy the goal identify problems, the *Professional* can *evaluate info*

manually or *use eSAP*. For each of these alternatives a weight has been assigned as shown in figure 2. The *use eSAP* task, for achieving the *Identify Problems* goal, has been assigned a value of 1. This is because, the use of the eSAP system means the Professional will not have to put much effort on the evaluation, since this will be processed automatically by the system. On the other hand, to satisfy the task *Evaluate Medical Info Manually*, the Professional will have to spend much effort to read, compare and evaluate the medical information. As a result, a value of 5 has been assigned. After the values of complexity for each goal and task have been assigned, the next step involves the identification of the contribution of each alternative to the other functional and security requirements of the actor. Such contributions are shown in figure 2 as dashed line links.

To denote the contributions of the different alternatives, we employ a quantitative approach presented by Giorgini et al [Gio02] according to which, each alternative provides a contribution between 0 and 1, where 0 means the alternative endangers the security or the functional requirement, whereas 1 means the alternative completely contributes towards the satisfaction of the security or the functional requirement. To keep the Figure simple, in this example we denote contributions to the *Keep Patient Anonymity* security constraint, only from the *Obtain OP Consent* secure goal alternatives (figure 2).

To calculate the total complexity with respect to an actor, the system complexity and the security complexity must be first calculated. The system complexity of an actor is given by adding all the weights assigned to the dependums that the actor has to satisfy. In addition, the security complexity of an actor is calculated by considering all the different options related to the security constraints weight of the actor. Then best suited option is chosen and the next step is to calculate the overall complexity for each actor. Table 2 indicates the different actors of the system and their system and security complexities.

Table 2: Complexity Values

Actor	System complexity	Security complexity
Older Person	4	0
Benefits Agency.	4	4
Professional	12	6
DoH	5	0
R&D Agency	0	0

The process of analysing the complexity and criticality with respect to security is based on the following procedure. Firstly we calculate, for each actor involved, the complexity (taking into account time) and the criticality (as indicated previously). The next step involves the insertion of actors who assume a greater value of complexity and criticality, than the maximum value they can assume, into two lists, comp-actorList and crit-actorList respectively. The process ends with the assignment of some security constraints to different actors of the system in order to reduce the complexity or the criticality of the "overloaded" actors. For instance, as it can be seen from Table 2, the professional actor is overloaded with respect to complexity. To reduce the overload, the actor diagram can be revised. For instance, the responsibility of providing medical information for research can be assigned to the *Department of Health* (DoH) reducing the overall complexity of the *Professional* actor from 21 to 15 and the security criticality from 10 to 5.

5 Conclusions

In this paper we have presented an analysis for evaluating the degree of complexity and criticality of the actors of the system, with respect to security. Such an analysis provides a valuable process for the developers of information systems because it allows them to identify possible security bottlenecks.

In addition, our analysis helps to justify possible trade offs between security and functional requirements. By knowing how critical an actor is with respect to security a decision can be made. Our aim is to provide a clear well guided process of integrating security and functional requirements throughout the whole range of the development stages. The ability to identify the bottlenecks of an information system with respect to security and justify the decisions behind possible trade offs between security and functional requirements can definitely help towards this aim.

This work is an ongoing research. The presented analysis covers only the requirements stage of the Tropos methodology. We are working towards extending our analysis to the next stages of the methodology, since such an analysis can help in the later stages of the development. For example, criticality and complexity can help us decide for different architectural choices during the architectural design stage of the methodology, such as the choice between mobile and static agents.

References

[Bre02] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos and A. Perini. TROPOS: An Agent Oriented Software Development Methodology. Submitted to the Journal of Autonomous Agents and Multi-Agent Systems. Kluwer Academic Publishers.

[Dev00] P. Devanbu, S. Stubblebine, "Software Engineering for Security: a Roadmap", Proceedings of the conference of The future of Software engineering, 2000.

[Gar02] M. Garzetti, P. Giorgini, J. Mylopoulos, F. Sannicolo, "Applying Tropos Methodology to a real case study: Complexity and Criticality Analysis", in the Proceedings of the Second Italian workshop on "WOA 2002 dagli oggetti agli agenti dall'informazione alla conoscenza", Milano, 18-19 November 2002.

[Gio02] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, R. Sebastiani. "Reasoning with Goal Models", in the Proceedings of the 21st International Conference on Conceptual Modeling (ER2002), Tampere, Finland, October 2002.

[Mou03] H. Mouratidis, P. Giorgini, G. Manson, "Modelling Secure Multiagent Systems", in the Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne-Australia, pp. 859-866, ACM 2003.

[Mou03b] H. Mouratidis, Secure Tropos: internal report. University of Sheffield, Department of Computer Science, September 2003.

[Mou03c] H. Mouratidis, I. Philp, G. Manson, "A Novel Agent-Based System to Support the Single Assessment Process for Older People", in the Journal of Health Informatics (9) 3, pp. 149-163, September 2003.

[Yu95] E. Yu, Modelling Strategic Relationships for Process Reengineering, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.