Master Thesis
Software Engineering
Thesis no: MSE-2007:05
January 2006

Reviewing and Evaluating Techniques for Modeling and Analyzing Security Requirements

Khalil Abu-Sheikh

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Khalil Abu Sheikh
Address: Kungsmarksvägen 1313
371 44 Karlskrona
E-mail: khaleel.abu.sheikh@gmail.com

University advisor(s):
Per Jönsson
Department of Systems and Software Engineering
School of Engineering
Blekinge Institute of Technology

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : +46 457 271 25

ABSTRACT

The software engineering community recognized the importance of addressing security requirements with other functional requirements from the beginning of the software development life cycle. Therefore, there are some techniques that have been developed to achieve this goal. Thus, we conducted a theoretical study that focuses on reviewing and evaluating some of the techniques that are used to model and analyze security requirements. Thus, the Abuse Cases, Misuse Cases, Data Sensitivity and Threat Analyses, Strategic Modeling, and Attack Trees techniques are investigated in detail to understand and highlight the similarities and differences between them. We found that using these techniques, in general, help requirements engineer to specify more detailed security requirements. Also, all of these techniques cover the concepts of security but in different levels. In addition, the existence of different techniques provides a variety of levels for modeling and analyzing security requirements. This helps requirements engineer to decide which technique to use in order to address security issues for the system under investigation. Finally, we found that using only one of these techniques will not be suitable enough to satisfy the security requirements of the system under investigation. Consequently, we consider that it would be beneficial to combine the Abuse Cases or Misuse Cases techniques with the Attack Trees technique or to combine the Strategic Modeling and Attack Trees techniques together in order to model and analyze security requirements of the system under investigation. The concentration on using the Attack Trees technique is due to the reusability of the produced attack trees, also this technique helps in covering a wide range of attacks, thus covering security concepts as well as security requirements in a proper way.

Keywords: Security Requirements, Abuse Cases, Misuse Cases, Data Sensitivity and Threat Analyses, Strategic Modeling, Attack Trees.

ACKNOWLEDGEMENTS

I would like to heartily acknowledge my advisor Per Jönsson for continuous encouragement during the extended time of writing this thesis. His guidance, professional style and valuable comments and recommendations helped me to accomplish this thesis on time. To my family, who gave me invaluable support over the years. Your encouragement is greatly appreciated. A special thanks to my friends who supported me during writing the thesis as well as reviewing and discussing some issues.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

This chapter introduces and presents the baseline of this master thesis to provide the reader with an overview on the whole paper and the important issues that will be discussed and investigated.

1.1 BACKGROUND AND MOTIVATION

Security is considered as one of the key factors for the success of many software systems; especially the ones that run under the Internet environment. The increased use of Internet leads to more and more attacks, threats and vulnerabilities. Providing a definition for security is not an easy task since the security idiom is used within various fields such as bank, police, IT etc. Software security, the focus of this thesis, is concerned with protecting the system and its resources from malicious access. Generally, information, especially sensitive information, is an important asset within any software and it should be protected. Information security is defined as *"the protection of information assets from a wide range of threats"* [1]. System security should be engineered in a systematic way to reach the required goal. Anderson [2] defines security engineering as *"security engineering is about building systems to remain dependable in the face of malice, error, or mischance. As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves"*. We prefer this definition because security has a dynamic form and this definition focus on the flexibility issues of the security engineering.

Nowadays secure systems are of greater interest than ever before. Devanbu et al. [3] summarize many issues when they said, *"Is there such a thing anymore as a software system that doesn't need to be secure?"*. Many security requirements are considered as critical requirements. In that sense, failure in fulfilling these requirements may endanger human life (as with aircraft or car control system), cause economic damage (cash machines) etc. Therefore; every software system must have defense mechanisms against malicious adversaries. In order to build strong enough security systems, security requirements must be involved in the software development process from the beginning. Thus, designing secure systems should be carried out by involving security requirements and threats in a systematic way that can be integrated with the conventional requirements engineering process. Unfortunately, security is often added as an afterthought [1] [3] [4] [5], this is one of the main reasons for security failure, another reason is that designers protect the wrong things, or protect the right things but in a wrong way.

Secure software is becoming increasingly indispensable; this is caused by the increasing use of the Internet [5] [6] which is not secure anymore [7]. Nowadays, networked-based distributed computer systems are widely used to store and manage highly sensitive information [8]. Using Internet applications increases the challenges and difficulties to build secure systems. Moreover, management looks forward to ensuring that the used security mechanism is strong enough and the sensitive information cannot be misused. This clarifies the intensive need for developing a capable system with credible defenses mechanisms, so software engineers need to have a deeper understanding of the security attacks, security mechanisms, and security services.

In order to deal with security requirements smoothly, it is necessary to take a decision for classifying security requirements as functional and non-functional requirements. Many researchers like [6] [8] [11] [12] consider and classify, in general, security requirements as

non-functional requirements. On the other hand, Parnas et al. [13] classify some security requirements as functional requirements. Chapter 4 explains in detail about this classification.

Building and designing secure systems require knowledge of security concepts. The main aspects for security are Confidentiality, Integrity and Availability (hereafter denoted as CIA) [1]. *Confidentiality* is about maintaining privacy (prevention of unauthorized disclosure of information). *Integrity* is about ensuring the accuracy and completeness of information (prevention of unauthorized modification of information). *Availability* is about protecting the system as well as its data to ensure it is available when required. At the same time, focusing on one aspect of security may not satisfy other aspects. By understanding these concepts, we can recognize that using only firewall and simple password authentication, which is widely used within many organizations, without understanding of specific security requirements does not provide adequate protection within the specific context [5] (e.g. using firewall and simple password authentication cannot prevent from denial of service attacks (DOS) that threats the system availability).

Building secure systems is considered as a main challenge for the software engineering community. Another significant challenge faced by the software engineering community is to develop the required product within the time limit, budget and personnel constraints. Thus, competitive software providers try to utilize the available resources to deliver a valuable product to their customers as early as possible. Another main challenge is the modeling and the analysis of security requirements [1] [3] [15] which is the focus of this thesis (*reviewing and evaluating the model and the analysis techniques for security requirements*). Moreover, the main contribution of the study as a whole is a clear focus on describing different techniques used to model and analyze security requirements. Thus, increasing the understanding of how security requirements are modeled and analyzed in the software development. This understanding helps in modeling and analyzing security requirements from the beginning of the software development life cycle; thus recognizing the way and the possibility of fitting these security requirements into the development process of the software under consideration. Also, it leads to prevent or at least minimize conflicts with other functional and non-functional requirements. Moreover, we can achieve significant benefits from analyzing and understanding security requirements in an effective way. These benefits include and are not limited to:

- Detecting and removing defects and conflicts earlier.
- Reducing development time and cost.
- Improving system quality.

Clearly, there is much more to be gained by understanding the modeling and analysis processes of security requirements.

1.2 THE RESEARCH

The study aims to investigate the modeling techniques as well as the analysis techniques of security requirements. Thus, a discussion of different modeling and analysis techniques will be conducted in order to investigate and get high level of understanding about their mechanisms. The main objectives are to identify and evaluate different modeling and analysis techniques.

The following research questions were considered for the thesis:

- How can security requirements be specified, modeled and analyzed?
- What are the differences between different techniques of modeling and analyzing security requirements?
- What are the benefits to have different techniques of modeling and analyzing security requirements?
- Do these techniques cover all the concepts of security?

First of all, we expect to cover the questions that mentioned previously. In addition, the following outcomes are expected:

- A comparison between some of the techniques that are used to model and analyze security requirements.
- Providing an explanation, whether some of these techniques or all of them are considered as suitable to fit security requirements into the conventional requirements engineering process. If none of them are suitable, suggestions for alternative(s) will be presented.
- A general suggestions and recommendations regarding the software development life cycle with emphasize on the related issues of the security process and involved participants.
- Whether there are missing things in these techniques?

The remainder of the thesis is structured as follows. Chapter 2 presents a set of related work; this related work will be classified into main categories. Chapter 3 discusses the research methodology that has been followed during the research. Chapter 4 holds a discussion about security requirements engineering for further understanding. Chapter 5 includes investigation on a set of modeling and analysis techniques for security requirements. Chapter 6 holds discussions and motivation. Chapter 7 holds the conclusions and finally Chapter 8 includes the future work.

Chapter 2

RELATED WORK

This section discusses some of the research that has been done on modeling and analyzing security requirements. We classify these techniques in specific categories according to the way they deal with the security requirements. Some of these techniques are designed to deal with both modeling and analyzing security requirements, while some of them can deal with either modeling or analyzing security requirements.

Liu et al. proposed a framework for dealing with security and privacy based on *I** framework, an agent-oriented requirements modeling language [16]. The main purpose was to define a set of analysis mechanisms for security and privacy aspects and integrate them into the conventional requirements engineering process. Therefore, security and privacy can be taken into account at the beginning of software development life cycle. A set of analysis techniques have been used to cover the most important issues related to security; the proposed analysis techniques are:

- Attacker analysis is used to identify potential system abusers and their malicious intents.
- Dependency vulnerability analysis aims to detect vulnerabilities that are caused by relationships among stakeholders.
- Countermeasure analysis supports system designers to take the right decision to protect the system under investigation from general potential attackers, vulnerabilities and threats.
- Access control analysis is used to refine the proposed solution and bring it closer to a system design which "*bridges the gap between security requirements models and security implementation models*" [16].

The proposed framework is designed to cover both analysis and modeling aspects of security requirements. It is very similar to the Strategic Modeling technique [6] (see section 5.3) since it is built upon that the Strategic Modeling technique where both of them are developed by the same team.

Abuse Cases technique is another technique that can be used to elicit, analyze and model security requirements [17]. Abuse cases demonstrate how negative scenarios can be used. McDermott et al. [17] confirm that mathematical solutions for security problems are hard to understand by persons who are not security specialists. Thus, they claim that using the Abuse Cases technique to analyze security requirements is simple, from a user perspective. Abuse case models, which are used to represent abuse cases, are similar to use cases; they can be documented using the same strategy as use cases by either diagrams built using the Unified Modeling Language (UML) notations or natural language. The main benefit of using the Abuse Cases technique is that it provides higher level of understanding for developers, analysts, designers etc. than mathematical security models. The Abuse Cases technique is more about modeling security requirements; in addition, using this technique provides acceptable level of understanding to analyze and deal with security requirements.

Data sensitivity is an important aspect that is mentioned and put forward by Kis [5] for analyzing security requirements. His work is concerned with understanding the real value of data that needs to be protected. Therefore, he suggests the use of Data Sensitivity and Threat Analyses technique. This idea helps in saving time and effort that might be wasted in protecting unimportant data, thereby providing strong enough protection for the most important information. Dealing with requirements is done as antipatterns which are observed

cases that help in recognizing and avoiding some common security pitfalls. Each antipattern can be presented using some elements, which are:

- A description of the problem and some relevant background.
- A description of the analysis of the context in which the problem usually arises and the faulty beliefs that lead to the antipattern solution.
- A description of the antipattern solution and the analysis of its security impact.
- A presentation of the symptoms of diagnosing the antipattern.

A detailed description and explanation example about using the Data Sensitivity and Threat Analyses technique will be discussed in section 5.2.

Sindre et al. define the concept of the Misuse Cases technique [18] which is similar to the Abuse Cases technique and both of them are used to elicit, model and analyze security requirements. The technique by Sindre et al. provides the ability to look into the functional requirements and security requirements together [18]. Therefore, they make an extension for the Use Cases technique to elicit, model and analyze security requirements. They define a misuse case as a special kind of use case, and it describes behaviors that the system should prevent its occurrence. The technique suggests presenting both use cases and misuse cases together in the same diagram. In order to make a differentiation between them, they depicted the misuse cases (and any mis-actors) in an inverted format as indicated in Figure 1, where black ovals and black actors represent misuse cases and mis-actors respectively, while white ovals and white actors represent use cases and their actors. UML is used as notation, but new kinds of relations have been defined and used which are *"detects"* and *"prevents"*, where the meaning of these relations is the same as the names suggest. Finally, they provide a step-by-step method that consists of five steps in order to build the diagram for both functional and security requirements. More detailed discussion about the Abuse Cases technique as well as the Misuse Cases technique will be carried out in section 5.1.

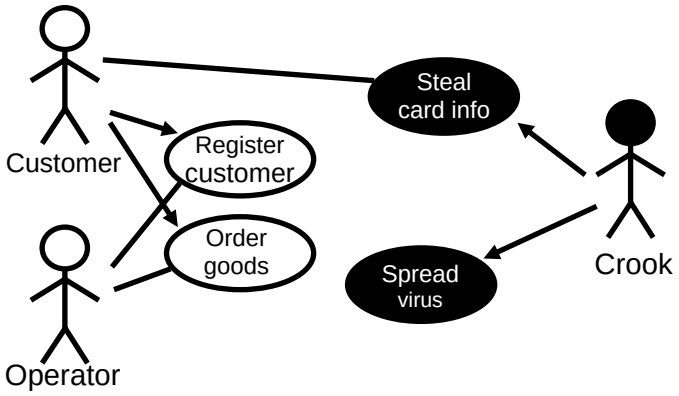


Figure 1: Misuse and mis-actors inverted [18].

Another work about non-functional requirements arena have been carried out by Chung et al. [7]. They classify security requirement as non-functional requirements and present a general framework to deal with non-functional requirements to express them explicitly. They believe that non-functional requirements are often subjective (it can be viewed and evaluated differently by different groups of people) and relative (it can be achieved in different ways). They introduced a set of sub-goals in order to satisfy a given security goal where the relationship between the sub-goals and the goal is either AND or OR relationships.

- *AND relationship* means that all soft-goals/sub-goals should be accomplished in order to satisfy the security goal
- *OR relationship* means that at least one of soft-goals should be accomplished in order to satisfy the security goal.

A *Soft-goal r*epresents a goal that has no clear-cut definition and/or criteria as to whether it is satisfied or not [7]. Thus, soft-goals are used to present security requirements or any

other nonfunctional requirements. Simply, they use the idea of divide and conquer to simplify security goals in order to deal with them in simple manner. Formally, the framework consists of five major components which are soft-goals, contributions, methods, correlation rules and evaluation procedure. Goals and soft-goals can be presented in a graphical diagram as well as in text format (in this case, zero or more parameters whose nature depends on the soft-goals might be used). The proposed framework helps in both modeling and analyzing security requirements. Figure 2 shows the representation modeling for the security of an account but in a high level representation. Thus, in order to satisfy the security of the account, one can decompose the security soft-goal into integrity, confidentiality and availability soft-goals. Then, these soft-goals can be refined to produce more specific soft-goals (e.g. confidentiality needs to be enforced for both information that reside inside the projected system, and information that will be in external media, such as backup files [7]). Graphically, an AND relationship is represented by a curved line that connects the lines drawn from one soft-goal to its subcomponents while OR relationship is represented by a double curved line that connects the lines drawn from one soft-goal to its subcomponents. However, it is worth to mention that many of the notations that are used within this framework are similar to the one used in I* framework (see section 5.3.2) such as equal, hurt, break, unknown, help, make etc.

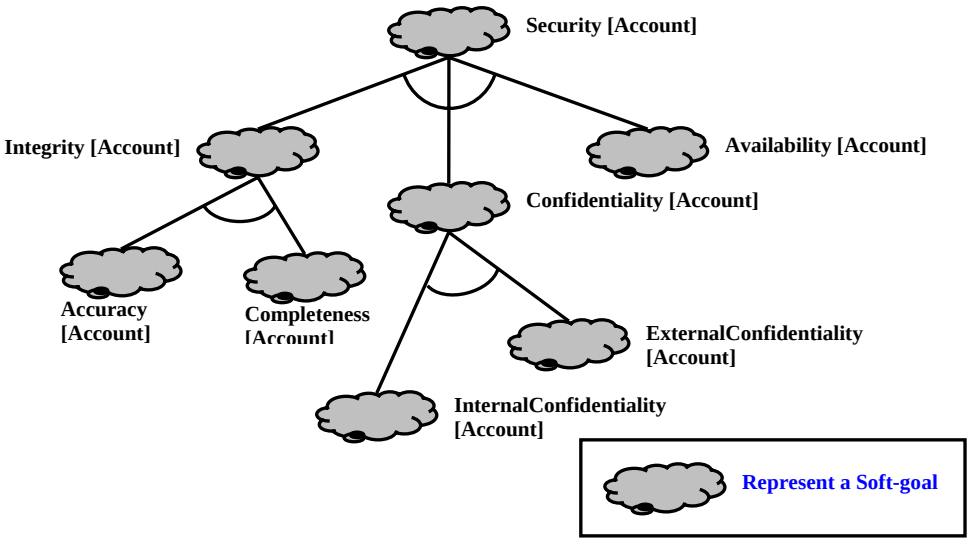


Figure 2: Representation for the Security Soft-goal [7]

Attack Trees is another technique that might be used to model and analyze security requirements. This technique has been developed by Bruce Schneier [49]. Attack trees are used to describe the security of the systems, based on varying attacks. The root node of an attack tree represents the goal of a specific attack. There are different ways to achieve this goal which are described in the children nodes of the root node. Thus, each child node is considered as a sub goal in the tree. Each path from a leaf node to the root node represents a possible attack against the goal. Any attack tree consists of two types of nodes: AND-nodes and OR-nodes. An Or-node represents the alternative ways to achieve the goal while an AND-Node represents different required actions to achieve the same goal. Once the attack trees are built, the analyst can assign values (Boolean, numerical or combinations of them) to the leaf nodes. The values of the AND and OR nodes are calculated according to the values of their child nodes. More detail and explanation of the Attack Trees technique can be found in section 5.4.

Lamsweerde et al. built their framework based on integrating intentional anti-goals, which are setup by attackers to break security goals, into the software development life cycle [19]. The proposed framework is similar to the Attack Trees technique. The *anti-goals* represent the software vulnerabilities that can be observable by the attackers or anti-requirements that can be implemented by them. The main goal of this proposed framework is to generate and resolve obstacles (or "*anti-goals*") as much as possible. In order to achieve that goal, attack

trees should be derived systematically by refining the anti-goals until the leaf nodes are reached. Thereby, new security requirements can be generated to solve the new attack vulnerabilities which are represented by anti-goals. The goal-oriented requirements engineering framework is used to present security requirements as *goals*. Security goals are connected to each other by AND/OR relationships where satisfying each goal depends on AND/OR relations, which have the same concepts as suggested by Chung et al. [7]. The proposed framework uses semiformal keywords like *Achieve*, *Avoid* and *Maintain* to name the system goals according to their intention (e.g. Goal Avoid [SensitiveInfoKnownByUnauthorizedAgent] [19]). The goals might be represented by UML class diagrams. Analyzing the security and refining the built attack trees can be carried out by using obstacle analysis which aims to identify the different ways of breaking the system goals in order to resolve each of them (i.e. resolve the generated anti-goals by defining the anti-requirements that can cover them). This framework can be used to model security requirements, also deriving attack trees help in understanding and analyzing security requirements deeply.

The Strategic Modeling technique is another technique that is suggested by Liu et al. [6] to model relationships among strategic actors in order to elicit, identify and analyze security requirements. They believe that security issues are ultimately about relationship among social actors – stakeholders, users, potential attackers, etc., and the software acting on their behalf. Therefore, I* framework is used to model relationships between strategic actors, where I* framework offers three basic types of concept for modeling application which are actors, intentional elements and links. In order to model the complex relationships among actors, they defined new concepts of agent and roles, considered as special cases of actors. A security goal might be divided into soft-goals which are organized using AND/OR relationships which have the same meaning as mentioned by Chung et al. [7]. Also, satisfying security goals depends on the AND/OR relationships, which have the same concepts as suggested by Chung et al. [7]. A set of analysis techniques are put forward which are [6]:

- Actor dependency analysis is used to identify attackers and their potential threats
- Actor goal analysis helps to elicit the dynamic decision making process of system players for security issues.
- Agent-oriented analysis is based on the idea of modeling and reusing abstract role patterns.
- Goal-oriented analysis is based on the idea that relationships on various kinds of requirements can come from generic knowledge organized in catalogues.
- Scenario-based analysis can be considered as an elaboration of the previous two kinds of analysis.

Finally, they claim that this technique is particularly suitable for new Internet applications with complex security challenges. More details about this technique can be found in section 5.3.

Cheng et al. [21] present a work for using security patterns to model and analyze security requirements. They believe that there are knowledge gaps among developers and the common approach to overcome these gaps is by using patterns (e.g. analysis patterns, design patterns, etc). During their work they described a suitable template for security patterns that is tailored to meet the needs of secure systems development. The presented template is similar to that used to describe design patterns [20] with some modifications to enhance the presentation of security and requirements-oriented information in order to facilitate reuse of security knowledge. Therefore, they altered several of the fields in the design pattern template. Also, additional information is added to the new patterns to include behavior, constraints, and related security principles that address difficulties inherent to the development of security-critical systems. To maximize comprehensibility, UML is used as notations to present structural and behavioral information. This template contains many

fields like Pattern Name and Classification, Intent (also known as Motivation), Applicability, Structure, Participants, and Collaborations which are taken from design patterns. Also, Cheng et al. added some additional fields to the template such as Behavior, Constraints, and Supported Security Principles. Applicability and Consequences are examples of the altered fields in order to provide and present more relevant information regarding security than the original template which describe design patterns. For instance, they altered the Applicability field to determine whether a pattern is applicable to a system by describing the circumstances and the assumptions that make the pattern suitable to be used.

Another relevant work is carried out by Giorgini et al., where they propose a framework for modeling and analyzing security and trust requirements that extend the Tropos methodology for early requirements modeling [22]. *Tropos framework* is *"an agent-oriented software development methodology, tailored to describe both the organization and the system"* [22]. Giorgini et al. mentioned that they identify clearly the roles of each actor who is responsible for manipulating resources, accomplishing goals or executing tasks, and actors that own or permit usage of resources or goals. They believe that security engineering should model the entire organization and procedures since security failures often are organizational or procedural failures. Tropos uses the concepts of [22]:

- *Actors* have strategic goals that represent agents (e.g. human being or a software/hardware system), roles (behavior of a social actor) or positions (set of roles).
- *A goal* represents the strategic interests of an actor.
- *A task* specifies a particular way of doing something (set of action that can be performed to satisfy a goal).
- *A resource* represents a physical or an informational entity.
- The depending actor is called the *depender* and the actor who is depended upon is called the *dependee* and the object which the dependency relationship centers around is called the *dependum*.
- *"A dependency between two actors means that one actor depends on another to accomplish a goal, execute a task, or deliver a resource"* [22].

Moreover, it is worth to mention that the I* is the modeling language of Tropos [44] [45] in which it supports the concepts of both goal and agent orientation. In addition, this propose framework is very similar to the Strategic Modeling technique (see section 5.3) suggested by Liu et al. [6], but since the work by Giorgini et al. deals with public key/trust management infrastructure, they identify four relationships to specify dependency between actors, which are [22]

- *Trust*, among two agents and a service,
- *Delegation*, among two agents and a service,
- *Ownership*, between an agent and a service, and
- *Offer*, between an agent and a service.

In general, we consider the proposed methodical frameworks by Liu et al. [6] [16], Chung et al. [7] and Giorgini et al. [22] in the same category named *Goal and Agent Orientation Category*, because all of these methodical frameworks are built on comparable concepts and the notation used to present them are almost similar; the I* framework notations can be considered as the shared feature between them. In addition, the Abuse Cases [17] and Misuse Cases [18] techniques are considered to be in the same category named *Abuse and Misuse Category* since both of them are built by the same logic (see section 5.1). Data Sensitivity and Threat Analyses [5] and Security Patterns [21] techniques are considered to be similar since both of them deal with the security requirements as patterns, thus we consider them in the same category named *Pattern Category*. Finally, Obstacles (anti-goals) [19] and Attack Trees [49] techniques are considered in the same category named *Attack Trees category*

since both of them involve the use of the attack trees to generate different kind of attacks regarding the system under investigation.

Due to the limited time, we can not review and discuss all of these techniques; thus we selected one technique from each of the previously mentioned categories to discuss in Chapter 5 except for the Abuse Cases and Misuse Cases techniques which belong to the same category. Selecting Abuse Cases and Misuse Cases techniques from the same category have been carried out as an example to show that there are some differences between them even if they are belong to the same category and they have similarities in concepts (see section 5.1.4.1). The selection of one technique from the other previously mentioned categories is carried out according to the following facts:

- *Goal and Agent Orientation Category*: the proposed methodical framework by Liu et al. [16] is built upon the Strategic Modeling technique [6], and it is considered as an expansion of the Strategic Modeling technique. Moreover, the other proposed methodical frameworks by Chung et al. [7] and Giorgini et al. [22] are similar to the Strategic Modeling technique. However, we found that the Strategic Modeling technique has less complication, thus we chose it among the others.
- *Pattern Category:* during gathering the related materials we noted that the idea of the Data Sensitivity and Threat Analyses technique [5] is very important and it should be discussed and investigated in detail in the thesis.
- *Attack Trees category:* we preferred to choose the Attack Trees technique since the other similar techniques such as the Obstacles (anti-goals) technique use the idea of the Attack Trees technique with more or less modification.

Finally, the Abuse Cases, the Misuse Cases, the Data Sensitivity and Threat Analyses, the Strategic Modeling and the Attack Trees techniques will be investigated and discussed in Chapter 5. Moreover, it is worth to mention that at the time of writing there were no similar surveys regarding these techniques have been conducted.

Chapter 3

RESEARCH METHODOLOGY

This chapter discusses the methodology of the research and provides necessary information to judge the validity of the thesis. It provides a precise description of how the study was done and explanation of how the results were analyzed.

3.1 RESEARCH METHOD

In order to address the research questions mentioned in section 1.2, a theoretical study has been conducted. The following subsections describe how the literature survey, analysis, reviewing and evaluating have been carried out.

3.1.1 LITERATURE SURVEY

The main reason of using a literature survey is that literature materials are widely available and easily accessible. So, a detailed and comprehensive literature survey has been carried out to gather relevant and related literature materials like books, articles, journals (e.g. requirements engineering journals) and conference proceedings regarding the topic at hand. Since the main purpose of the thesis is about reviews and evaluation, reading a lot of literature about chosen topic area is essential. In order to focus on the topic area and to save time and efforts, the survey should be conducted in a systematic way. Therefore, we managed the literature survey and limited it to the following keywords.

- Security requirements engineering
- Analyzing security requirements
- Modeling security requirements

Also, a combination of the words security, requirements, analysis and modeling were used to get relevant resources. In addition, we quickly scanned the reference list for each material we used hoping to find other relevant materials.

The search has been carried out using different kinds of databases which were IEEE Xplore, ACM digital library, and engineering village website which offers Compedex and Inspec databases. Also, different kinds of search engines (e.g. Google [52], Google scholar [53], yahoo [54] and ask [55]) with their advanced search functionality like searching for a specific file format, searching for recent published articles, searching for a phrase with/without specific word(s) etc. were used during the search procedure. The idea behind using different kinds of search engines was to generate a variety of outputs which helped to cover the most important relevant materials, since each search engine has its advantages and disadvantages. However, searching within the Internet cannot provide all the needed resources (i.e. not all printed books are available in electronic form, and if so, it might not be freely accessible), so we searched for printed materials within the university library.

3.1.2 ANALYSIS PROCEDURE

The first step after gathering the literature materials was to check whether it is really relevant; this was performed by carefully reading the abstract and the conclusion parts. Then, a focused reading for the chosen materials was carried out to decide whether it added new knowledge, better understanding or confirmed the facts that were found in other materials. This was done by reading each document once or more depending on its structure, size, used examples to explain a certain idea etc. Following these steps helped to understand the

purpose of different materials, idioms, notations (e.g. UML notations that are used in the Abuse Cases and Misuse Cases techniques) used during the descriptive examples. Moreover, the intention of each literature was tried to understood by taking notes about the important aspects in order to focus on them (e.g. how the technique worked, what things that the analysis is concerned about). Also, the main issues that were included in each material were documented briefly; thereby used as quick references. Furthermore, every element in the used materials was tried to understood and the reason of their existence. These processes helped to achieve fair enough understanding of the study area. In addition, our knowledge and experience of the security requirements area was used. This knowledge was built from

- The courses taken in the school
- Discussion of some issues with the school teachers and students
- Reading some relevant documents

The experience came from the previous work as developer and some experience as an analyst for system requirements. In order to increase the knowledge, different sources that describe the same technique were examined. Finally in order to improve the quality of the discussion, some materials that discussed the same topic but from different points of view or by using different analytic techniques were used

3.1.3 REVIEW AND EVALUATION PROCEDURE

During this procedure we tried to analyze, classify and evaluate the each of the techniques according to following issues:

- Whether the technique covers the security concepts (CIA).
- The strongest points in the technique.
- Overall performance of the technique.
- Possibility of using the technique in practice.
- Whether the technique depends on or uses trusted assumption.

Trusted assumption is a condition assumed to be true such as using Kerberos, a network authentication service originally developed for Project Athena at MIT [40], which is used to authenticate the users and their requests of services; another assumption could be that the attacker will use product XYZ to accomplish an attack process. Since the classification and evaluation were built according to our point of view and understanding, we tried to discuss and compare the techniques that have some similarity in concepts and/or process (e.g. Abuse Cases and Misuse Cases techniques) in more detail for better understanding. This procedure helped to discover diversities and similarities between different used techniques.

3.2 RESEARCH TAXONOMY

Research methods can be classified in various ways; however, one of the most common classifications is qualitative and quantitative research methods [23]. Qualitative research methods involve the use of qualitative data, such as interviews, documents, and observation data, to understand and explain a phenomenon. Qualitative researchers could be found in many disciplines and fields, using a variety of approaches, methods and techniques. Quantitative research methods were originally developed in the natural sciences to study natural phenomena. Examples of quantitative methods are survey methods, experiments and numerical methods such as mathematical modeling. Furthermore, some researchers have suggested combining one or more research method in one study called triangulation [24]. The research methodology used in this thesis is closer to qualitative research than quantitative; since the main source of the information is documents such as books and published articles.

According to Dawson [23], research can be classified from three different perspectives: its field, its approach, and its nature. The field of the research is *"little more than a labeling*

device that enables groups of researchers with similar interest to be identified" [23]. Therefore, this thesis can be categorized within the software engineering field. Research approach represents the methods that are used as a part of the research process; thus, this thesis uses a literature survey, so it can be classified as literature survey. Research nature is about type of contribution that research makes to knowledge depending upon its nature [23]. The main contribution of the thesis as a whole is investigating, reviewing and evaluating different techniques used to model and analyze security requirements.

3.3 RESEARCH VALIDITY

3.3.1 VALIDITY TYPES AND THEIR RELATIONS TO THE THESIS

According to Trochim [9], there are four validity types which are construct validity, internal validity, external validity and conclusion validity (also known as statistical conclusion validity). However, Golafshani [59] mentions that *"some qualitative researchers have argued that the term validity is not applicable to qualitative research, but at the same time, they have realized the need for some kind of qualifying check or measure for their research"*.

Moreover, Winter [58] mentioned that construct validity is concerned with testing whether the study measure what it is intended to measure. Thus, the construct validity is related more to the quantitative research than qualitative research. Thereby and since this thesis is classified as qualitative research, this type of validity is considered as irrelevant. Internal validity is concerned with the cause-effect or causal relationships between the treatment and the outcome [9] [58]. Internal validity can be used with the experimental research; however it is irrelevant to the thesis. External validity is related to generalizing the results of the research [9] [58]. It is applicable to the experiment research that involved the use of sample from a certain population. External validity is not relevant to the thesis since there are no generalizations carried out. Conclusion validity is *"the degrees to which conclusions we reach about relationships in our data are reasonable"* [9]. Moreover, Trochim [9] mentioned that the conclusion validity was originally considered to be a statistical inference issue (i.e. evaluating whether the data analysis procedure produce the correct answer), but it has become clearer that it is also relevant to the qualitative research. Thereby and since the conclusion is based on our interpretation and understanding of the techniques that are used to model and analyze security requirements, we think that it is necessarily to discuss the interpretation validity as well as quality validity for the thesis to improve the conclusion validity as a result. Moreover, it is worth to mention that both the interpretation validity and quality validity are highly related to each other (i.e. good interpretation produce high quality).

3.3.2 QUALITY VALIDITY

Missing some important literature sources is considered as a major threat for the quality of the thesis; we tried to overcome that by following structured and systematic search procedures as shown in section 3.1.1. The results would have been more credible if different ethnographic techniques like observation, interviews, etc. would have been applied. Using triangulation of data sources [24] is preferable, where one data source can validate another by comparing the sources and see if they are homogenous. In order to achieve that, we compare different literature sources to validate each other. However, using many sources requires more efforts and thereby it might not always be possible to follow this approach.

3.3.3 INTERPRETATION VALIDITY

Misunderstanding of some issues in the literature documents such as techniques processes, concepts etc. is considered as a major threat for the interpretation of one technique or more. In order to overcome that or at least minimize the possibility of misunderstanding we have referred to many researchers who discussed that issue from the same or different

point of view. Moreover, we tried to consult different books to gain deeper understanding as well as to obtain a clearer picture. As a result, we tried to understand the meaning of texts, signs and symbols in the language or framework that is used to model security requirements as well as the analysis process of the security requirements (see section 3.1.3).

Chapter 4

REQUIREMENTS ENGINEERING AND SECURITY REQUIREMENTS ENGINEERING

This chapter discusses some issues related to security requirements and how to deal with security requirements within the requirements engineering process.

Security requirements engineering might be considered as a branch of requirements engineering domain. In other words it is subset of requirements engineering where requirements engineering is concerned about finding out what the system is supposed to do and the environments it is intended to operate in [25] [26] [27]. Security requirements engineering is generally concerned about how to deal with security requirements within the system development life cycle. In addition, it deals with finding and specifying the behavior of the security requirements within the system where some of these requirements are considered as system constraints or constraints for other functional requirements.

In general, both security and non-security requirements are used during the development process. Therefore, what we need is to extend the conventional requirements engineering process to handle security requirements simultaneously with the functional requirements. In addition, the main idea is to merge the security requirements within the conventional requirements engineering process in order to save project time and personal efforts as well as handling security requirements from the beginning of the project development life cycle. Moreover, many of the functional requirements are highly related to the security requirements, therefore it is better to investigate both types at the same time within the same process. Take the e-bank system for instance; it is clear that such systems should be highly secured, however it also contains many functional requirements like *"it must be possible to add new account"*. In addition, some of the functional requirements like *"each user of the system must have a unique system user ID"* have a strong relationship with security requirements such as *"system should block the user account and ask him/her to contact the bank after three wrong login trials"*. Finally, in this chapter, details about the conventional requirements engineering process are true for the security requirements engineering process also, unless something else is mentioned.

In order to build secure systems, requirements engineers have to follow a structure process aiming to elicit and define the security requirements and their constraints for other functional requirements. Finding, specifying and understanding security requirements at the beginning of the development life cycle helps to prevent or at least minimize the re-work cost, effort and time.

Generally, requirements engineering is more than an engineering approach, actually requirements engineering is multi-disciplinary [28], and it comprises other domains like human, cognitive and social sciences in order to deal with the used techniques for requirements elicitation, analysis and modeling. Some of the fields that are used in the requirements engineering are cognitive and human psychology, anthropology, sociology and linguistics [26] [29]; where these fields are very helpful in understanding stakeholders' need as well as the system behavior.

4.1 REQUIREMENTS ENGINEERING PROCESS

Before starting the discussion of the requirements engineering process, it is necessary to have a definition for that process to understand its meaning and its scope. Sommerville [25] defined the requirements engineering process as *"the structured set of activities concerned*

with eliciting, analyzing and documenting the system requirements." This definition does not cover the management, validation, verification and the traceability of the requirements which should be taken into account during the requirements engineering process. On the other hand, Zave [28] provides a more detailed definition as follows: *"Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."*

The requirements engineering process is like any other structured process that comprises a set of organized activities. In reality, the requirements engineering process is always cyclic process (see Figure 3). Individual activities are repeated as the software requirements are derived and the iteration continues during system implementation. Thus, the developed system will contain some of the requirements which are the most important requirements for the stakeholders as well as for the system. However, other collected requirements might be developed in the next release of the system which depends on many factors like the importance of the requirements, conflict with other requirements, cost, needed time etc.

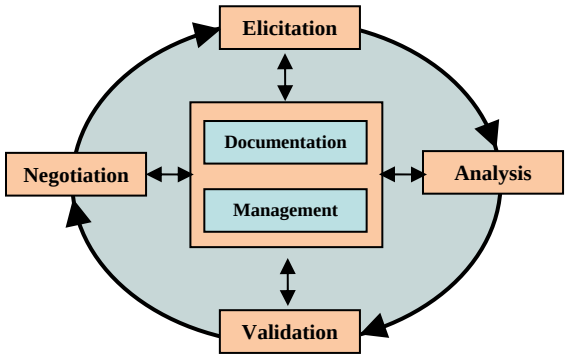


Figure 3: The requirements engineering activity cycle [27].

Requirements engineering activities have certain mechanisms, processes, information and knowledge. These activities transform inputs into one single output called the requirements specification or requirements document that defines what is to be implemented [27]. Also Software Requirements Specification (SRS) is another name given to such document [25] [28].

Developing a requirements specification for the security requirements is harder than developing a requirements specification for functional requirements. Relatively, eliciting and documenting functional requirements is easy since it is possible to know beforehand which services the system is going to provide, while on the other hand eliciting, documenting and managing security requirements is hard since it is difficult to determine exactly beforehand all the threats that can threat the system. However, the difficulty mentioned here is about specifying the security requirements in detail (i.e. eliciting the security requirements in a high level, then modeling and analyzing them to figure out a more detailed security requirements); for example, the requirements engineer may elicit and document in a high level security requirement such as *"each user account in the system should be protected"*, but after modeling and analyzing this requirement, he/she can find many detailed security requirements such as *"the system accounts should be available for the legitimate users all the time"*, *"the stored information in the users accounts should not be altered by anybody except the user himself"*, *"the users accounts should be accessible only by the legitimate users"* etc. Also, these goals might be decomposed into more detailed requirements. Moreover, sometimes it is difficult to describe the security requirements using only the natural language; thus requirements engineer may use diagrams like UML as a standard or any other kind of diagrams to provide more description and understanding for these requirements. Furthermore, Wilander et al. conducted a field study of eleven software projects including e-business, health care and military applications and they provide an overall conclusion which is *"security requirements are poorly specified due to three things: inconsistency in the*

selection of requirements, inconsistency in level of detail, and almost no requirements on standard security solutions" [35].

One important thing that should be kept in mind is that the requirements engineering process is different for different products and organizations, which indicates that the requirements engineering process depends on the type of products being developed, the size and culture of the companies involved and the software acquisition processes used [27]. Also, it is important to recognize that the requirements engineering process and its output are affected by and depends on the representative set of customers and stakeholders involved in the process [30] [29], since they have different backgrounds, experiences and needs [31].

4.2 REQUIREMENTS ENGINEERING PHASES

The requirements engineering process contains set of phases, these phases provide the ability to understand the customers' needs, define the system requirements and constraints, analyze them, evaluate their feasibility, determine customers' real need, validate requirements specification and manage the requirements. In order to understand the customers' real needs, the requirements engineer may conduct several meetings with different stakeholders and perform a pilot study to define the project goal(s) and the used terminologies; on the other hand he/she can determine customers' real need during the analysis process where he/she can check the necessity of the requirements by removing the unnecessary requirements and keeping the focus on real needs according to his/her thoughts and the necessity check already performed [25]. Also, prioritizing, evaluating and checking feasibility of the requirements can help in determining beneficial features that should be provided by the system; thereby developing a system that includes the real needs of the customers.

Describing these phases in detail is out of the scope of our topic, so we will mention the main principles for these phases briefly. The requirements engineering process consists of five main phases [25], which are:

- Requirements Elicitation
- Requirements Analysis and Negotiation
- Requirements Documentation
- Requirements Validation
- Requirements Management

The topic of this thesis will cover the first three phases for the security requirements within the requirements engineering process. Some of the techniques like the Misuse Cases and Abuse Cases techniques could be used to elicit and document the security requirements. Documenting security requirements using Abuse Cases or Misuse Cases techniques might be done using UML or natural language. The Strategic Modeling technique is one example for a technique that can be used to cover the analysis phase.

Requirements elicitation is the first phase of the requirements engineering process. It is normally known as the process of finding out what are the real needs of the customers as well as of the system [25]. It also includes activities to explore how the software can meet the stakeholders' goals and what alternatives might exist [30].

The analysis phase consists of a set of activities aimed to discover problems within the system requirements and achieve agreement on changes to satisfy all system stakeholders. Requirements analysis phase is overlapped with the elicitation phase when discovering a problem with elicited requirements. In other words, some analysis is carried out whenever requirements are discovered; thus problems within these requirements might be recognized immediately, discussed with the source and resolved [10]. Also, when the analyst discovers some problems with the requirements during the analysis phase, these requirements that have some problems can be sent back to the elicitation phase. This process is related to the requirements that are incomplete, ambiguous, conflict, etc. Negotiation phase is known as

"the process of discussing conflicts in requirements and finding some compromise which all of the stakeholders can live with" [25]. The principle of this process should be objective, where the judgments and the compromise for the system requirements should be based on technical and organizational needs. All the conflict requirements identified during the analysis process should be negotiated and discussed individually with the stakeholders in order to resolve the conflicts [25] [32].

Sommerville [25] defined the requirements document as *"the official statement of the system requirements for customers, end-users and software developers"*. Requirement documentation should be carried out with all activities within requirements engineering [32]. The requirements that are gathered during the elicitation phase should be recorded in the initial draft of requirements specification. This draft helps and provides the ability to build the product; also it helps in analysis and negotiation phase [33] [32] as well as in the management of the requirements. A good requirements document is unambiguous, complete, correct, understandable, consistent, concise and feasible. In addition to that, it should describe the service, functions, constraints, application domain, properties, etc that the software should satisfy [25] [30]. Documentation is normally written in natural language and it may also contain graphical models, mathematical models, prototypes or the combination of these methods. This helps the readers in gaining better understanding of the system. Sommerville [25] suggests the use of a standard template like, IEEE/ANSI 830-1993 in order to develop the requirements specification.

Requirements validation is the final stage of the requirements engineering process and is defined as *"the process of checking the requirements document for consistency, completeness and accuracy"* [25]. The main focus in this phase is on checking the final draft of requirements document for conflicts, omissions and deviations from different standards [25] also it is aimed to check if the requirements satisfy the user needs and intentions [27] [30]. To solve these problems, requirements engineer usually has to revisit the earlier process stages of the requirements elicitation, analysis and negotiation; however, in the worst case he/she may need to redo the elicitation, analysis and negotiation phases. It is very important to consider that requirements analysis phase and validation phase are overlapping as follow:

- Requirements analysis phase aims to see if we have got the right requirements. This can be done by negotiating the 'raw' requirements [25] with the system stakeholders. Some techniques that might be used are analysis checklist, interaction matrices etc.
- In contrast to those requirements, which are already analyzed, validation aims to check if we have the requirements right. This can be done by checking the final draft of the requirements document which includes all system requirements and where known incomplete and inconsistent requirements have been removed to decide whether the document and the requirements follow the defined quality standards [25]. Some techniques that might be used are requirements reviews, prototyping etc.

Consider the following example for a library system for more clarification; the customer mentions during the elicitation phase that *"the system should generate a letter each week for each student who has one or more overdue borrowed items (e.g. DVD, CD, book etc)"*. In the analysis phase, requirements engineer may ask the customer to confirm whether the previous requirement presents his/her real need. Also, requirements engineer may ask for more clarification regarding the previous requirement to overcome the incomplete and ambiguous issues such as whether there is a preferable day to generate the letter, whether there is a certain process that the system should follow to generate the letter (e.g. when the item became overdue the system should generate the letter automatically), what should the letter include? etc. In the validation phase, a requirements engineer who preferably was not involved in the elicitation phase [25] should check the final draft of the requirements document to find some problems (incompetence, ambiguity etc) that have not been discovered in the analysis phase. Also, requirements engineer may develop a sample paper based prototype for the letter that should be generated by the system in order to agree with the customer about the letter contents.

After performing the validation process the final requirements document is prepared and approved, which is used as the guideline to develop the system. The outputs to this process are a list of requirements problems and agreed upon actions in order to address these problems [25] [32]. Requirements validation is a very important activity since discovering and fixing requirements problems within this phase helps in avoiding a lot of expensive rework in design and implementation phases. Sommerville [25] mentioned that errors in delivered software systems which are a consequence of requirements errors may cost up to 100 times as much to repair as programming errors.

Requirements management is *"the process of managing changes to system's requirements."* [25]. It is an activity that is carried out in parallel to other requirements engineering activities till the last and final draft of requirements has been prepared [27] [28] [33] (see Figure 3). Also requirements management aims to ensure that all the requirements engineering activities are well organized and the final products are properly documented and traced [34]. In reality, this process will continue during the development process and after the deployment of the system. It is very important to keep in mind that requirements change is inevitable and does not mean that the requirements engineering process is poor [25] since there are various reasons behind the requirements change like, new emerged requirements, errors, conflicts or inconsistent requirements, change customer priorities etc [28]. Sommerville [25] claims that 50% or more of the requirements in general will be modified.

4.3 OTHER ISSUES RELATED TO SECURITY REQUIREMENTS

There is a great tricky question that requirements engineers should think carefully about which is: do we need to protect everything in the system? Absolutely, answering this question depends on the developed system and the environments it is intended to operate in. Clearly, everything should be protected but in different levels; for example personal information for customers should be protected in a higher level than information about offered products by an organization. In order to decide what thing should be protected and the level of the security, requirements engineers should understand and evaluate security requirements in order to prioritize them. Generally, requirements prioritization process is considered as one important factor that leads to building an effective system since this process helps in deciding the importance and the effectiveness of each requirement; thus deciding requirements that should be developed in the current version of the system. Therefore, prioritizing security requirements is highly related with the functionality that will be provided by the system. In order to prioritize security requirements efficiently, requirements engineers should collaborate in an efficient way with the system stakeholders who recognize the sensitivity of the information processed by the system. Also, requirements engineers should use their experience and available information about how hackers break system security; this may help in avoiding for example the DOS attack which occurs when the attacker flood the system with messages, service requests etc that cannot be handled by the system; thus making the system unavailable for its intended users.

Developing any system should be done with a tradeoff between security and other non-functional requirements such as performance, time to market [6] [3], usability etc. Performance might be considered as an important issue to any secure system; consider e-bank system for example, it is very clear that transferring money between different accounts or using VISAcard to purchase something from Internet should be done securely on a high level of performance. On the other hand, designing a truly and fully secure system might be too expensive in terms of cost and time and may affect the system performance since applying strong encryption mechanism for everything within the developed system will affect the system performance. In other words, encrypting and decrypting information usually is considered as time consuming process especially if the system is accessed by a large number of users and/or there is a large amount of information to be protected.

Another issue or threat related to security occurs when the bank sends an important document (e.g. PIN code) or VISAcard by mail to the customer address while at the same time the customer has changed his/her address without informing the bank about the new address. In this case, another person may have received the documents or VISAcard and he/she may use it. Security requirements have nothing to do with such a problem, but requirements engineer might check the bank process for such cases. This indicates that not all security failures are related to the security requirements engineering process, further to organization should review their process to be sure it is secure enough. An example of avoiding the problem is that an important documents or cards like VISAcard, Master-card etc should be taken from the bank personally by the customer himself/herself.

Social engineering is another main issue that is related to security and it is considered as one of the main threats to the systems. Social engineering occurs when someone pretends to be someone else (e.g. a person pretends to be a doctor to get some personal information about somebody). This is a smart way of attack since *"the attacker will extract the information directly, from people who are authorized to access it, by telling some plausible untruth"* [2] (e.g. the attacker might claim that this is an emergency call and he/she needs the information urgently). One possibility to avoid the problem is that the organizations should always ask for a secret key or an answer for special question before providing any information. Dealing with social engineering attack is not an easy task since this kind of attack is related more to the organization process than security requirements engineering. One way to avoid such cases is that security requirements engineers should mention such cases to the owner(s) of the system. Also, organization should review their process with security experts to prevent or at least to minimize the damage that might be caused by a social engineering attack.

Classifying security requirements as functional or non-functional requirements has got a great attention in the requirements engineering research field. Mylopoulos et al. [11] and Lamsweerde et al. [12] have stated broadly that all security requirements are non-functional requirements. Also, Chung et al. [7] classify security requirement, in general, as non-functional requirements. In addition, requirements that are related to the security concepts such as availability, authentication and privacy are considered as non-functional requirements [6]. On the other hand, Parnas et al. [13] mentioned that some security requirements can be considered as functional requirements where they argued that functional requirements are about state changing; thus, functional requirements define how those functions will behave and what states can be reached. One example to clarify this issue is the password authentication; the requirement *"the system should block the login for 15 minutes after three wrong entries"* is a state changing requirement, thus it is functional requirements; while the correct number of wrong entries is non-functional requirement because state does not change. Our vision is similar to Parnas et al. [13]. In addition, high-level non-functional requirements can be decomposed into functional system requirements [10]; this implies that security requirements might be considered as functional requirements. Anyhow, we believe that it is very difficult to classify security requirements as functional and non-functional requirements. Thereby, it is possible to consider some security requirements as functional requirements.

Architecture and design are other main issues that have significant effects in building secure systems. Zhang [36] and Jürjens [37] claim that the non-functional properties (e.g. security, performance, reliability etc) inside the software architecture are very poorly understood and still insufficient. However, software designers have recognized the need to incorporate non-functional considerations like performance and reliability into software design processes since they understood very well that adding performance and reliability requirements into software architectures after the fact is difficult or impossible [3]. Unfortunately, security requirements are not taken into consideration in advance: very often, security is an afterthought [3] [5] [36]. For that reason, adding security requirements into preexisting design will lead to serious (sometimes impossible) design challenges for the enforcement mechanism and the rest of the system [3] since it might require changing the

whole system design to be compatible with the security framework and its environment. Also, adding security requirements into preexisting design might be considered as a process that is expensive and difficult to handle. In addition, some application areas, like health and banking, have extremely strong requirements for non-functional properties [36] and the failure in satisfying those nonfunctional properties in any stage of the development life cycle process (i.e. the requirements engineering process, design, architecture, development, testing etc) may bring serious loss.

However, the advent of networking and open standards often provide new business reasons to re-engineer internal legacy systems (which are operated within secure intra-nets) for operation over the open Internet [3]. In such cases, there is no alternative to add security to a system after initial design and development. One of the specific issues that is addressed by Devanbu et al. regarding re-engineering of internal legacy systems is the *"Legacy Security Mismatches"* which is known as a mismatch between the security framework in the legacy system and the security framework of the target standard protocol (e.g. Unix systems and CORBA have different security policies and enforcement mechanisms) [3].

In addition and since requirements team is often not qualified to make architecture decisions; requirements engineer should be careful when describing security requirements to avoid unnecessarily and prematurely specifying architectural mechanisms for fulfilling security requirements (e.g., specifying the use of user identifiers and passwords as authentication requirements) [35]. Thus, system architect should have the possibility to choose the most appropriate security mechanisms (e.g. fingerprints, access card, iris pattern etc) to meet the core of the security requirements. If specific security architectural mechanisms and designs must be specified for legal, contractual reasons etc, then requirements engineer has to specify them as architectural and design constraints, not as security requirements [35].

Thus, we need to refine security requirements and design processes to focus on the security issue from the beginning. Jürjens tried to improve the relation between security requirements and software architecture by integrating security requirements analysis with a standard development process by extending UML, called UMLsec, to include modeling of security related features such as confidentiality, access control etc. The main reasons behind choosing UML and extending it to UMLsec are to make it available to developers who may not be specialized in security, but who have experience in software modeling using UML notations as well as considering security from the beginning in the system design stage.

Chapter 5
TECHNIQUES FOR MODELING AND ANALYZING SECURITY REQUIREMENTS

In this chapter we try to investigate and discuss different sorts of techniques used for modeling and analyzing security requirements in order to review and evaluate them. The techniques that will be investigated are:

- The Abuse Cases technique (see section 5.1.2).
- The Misuse Cases technique (see section 5.1.3).
- The Data Sensitivity and Threat Analyses technique (see section 5.2).
- The Strategic Modeling technique (see section 5.3).
- The Attack Trees technique (see section 5.4).

Illustrative examples within different domains have been used to provide better understanding and to present a complete picture of these techniques.

5.1 ABUSE CASES AND MISUSE CASES TECHNIQUES

Both the Abuse Cases and Misuse Cases techniques are derived from the Use Cases technique which is widely used to elicit and analyze functional requirements. The main purpose behind developing the Abuse Cases and Misuse Cases techniques was to provide simple techniques that offers an easy way to elicit, model and analyze security requirements; thus having better understanding and developing them within effective software systems.

Before discussing the Abuse Cases and Misuse Cases techniques, it is very important to introduce the Use Cases technique since both the Abuse Case and Misuse Cases techniques inherit many features from it. Therefore, the next subsections hold a discussion about the Use Cases, Abuse Cases and Misuse Cases techniques.

5.1.1 USE CASES TECHNIQUE

Use case is known as *"abstract episodes of interaction between a system and its environment"* [17]. Use cases are used to describe the complete interaction between the system and its environment, components or one or more of its actors. *Actor* is an external agent or a subsystem that interacts with the system and could be human, organization or physical agent such as printer, modem etc. Each use case must be complete in terms of describing the interaction between the system and its actors.

Documenting *use case models* can be carried out by using use case diagrams and/or use case descriptions. *Use case diagrams* are part of the UML and *use case descriptions* can be performed using natural language since there is no standard notation for the use case descriptions [17]. But it is highly recommended to document the use case models using both diagrams and natural language description. Thus, each use case diagram should have a use case description or scenario that shows and explains how the actor(s) and the system will interact. *A scenario* is *"a description of a specific interaction between particular individuals"* [17]. Figure 4 shows the used symbols for building a use case diagram. Each actor is represented by a stick figure; its name appears below the stick figure. Each use case is represented by an oval; its name appears either below or within the oval. Finally, the association line which connects an actor with a use case is represented by a line which

optionally might include arrowhead at one end of the line. It is worth to mention that an actor might be connected to one use case or more.

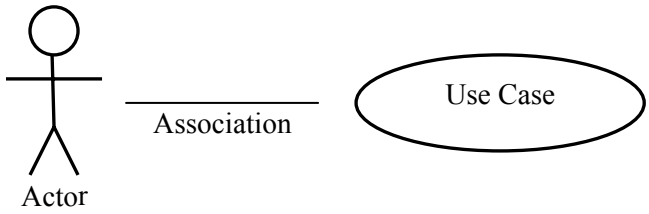


Figure 4: Use case diagram

To gain a clear picture of the use case model, we develop a simple use case model for a school mailing system (hereafter denoted as *Mailing System*) in order to show and clarify how actors, use cases, associations, diagrams and use case descriptions fit together. This use case model will be used to build the abuse case and misuse case models for the same Mailing System. In addition, the Mailing System use case model will be linked to the abuse case and misuse case models to get an overview about the relationship between the use case, abuse case and misuse case models.

5.1.1.1 USE CASE DIAGRAMS

Before drawing the use case diagram, it is very important to take a deep look at the case under investigation in order to find and decide the actors and their activities that will be used to draw the use case diagram. In the Mailing System example, we focused on the User as the main actor. Figure 5 shows the simple use case diagram for the Mailing System.

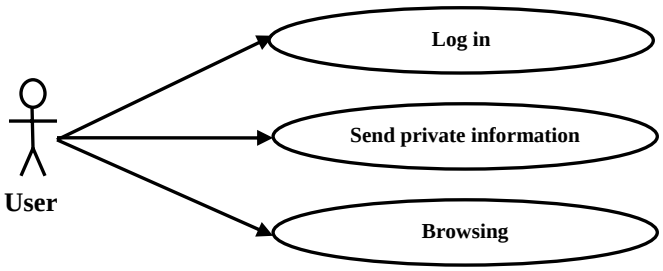


Figure 5: Use case diagram for a school mailing system.

We can notice that the use case diagram which shown in Figure 5 does not provide the full picture for the relation between the use cases and the actor. Therefore, use case description is needed in order to give the right meaning.

5.1.1.2 USE CASE DESCRIPTIONS

Recalling that the natural language should be used to document every use case within the corresponding use case diagram to shows how the actor(s) and the system will interact. For example, the comprehensive description of the use case "*Login*" in the use case diagram shown in Figure 4 might be as follows: *"System asks the User to enter his/her username and password in order to login to the system. The User enters his/her username and password, if they were incorrect; the system displays an error message and asks the User to try again. The User has three trials and after the third failure, the system locks the User account for twenty minutes. If the User forgets his/her username or the password he/she can ask the system to send this information to his/her email account, accordingly he/she should enter a valid email; otherwise the system displays an error message indicated that the provided email is not correct. If the User enters his/her username and password correctly, the system presents his/her account page"*.

Now and after explaining the use case model, diagram and description where all of them together compose the Use Cases technique; we can start investigating the Abuse Cases and Misuse Cases techniques as shown in the next subsections.

5.1.2 ABUSE CASES TECHNIQUE

Abuse case is known as the abstract episodes of interaction between the system and its environment, components or one or more of its actors, where the results of this interaction are harmful to the system, one or more of the actors or one or more of the system stakeholders [17]. Thus, this definition of the abuse case was built in terms of interactions that result in actual harm to the system, actors or stakeholders of the system. The above definition shows that Abuse Cases technique is a technique for eliciting and modeling security requirements; in other words defining the harmful interactions in order to prevent or avoid them in the system under consideration. Therefore, the type of the interaction and its result are used to distinguish between the use case and the abuse case models.

The idea of the abuse cases comes from abusing the system in someway. Abusing system might occur in different forms such as abusing privilege by some system agents, external malicious agents trying to gain some private information, unintentional abusing etc. Anyhow, requirements engineer should think about all possibilities for affecting the system negatively (causing harm to the system) while building abuse case models.

5.1.2.1 ABUSE CASES PROCESS

Many issues should be taken into consideration like amount of description, different ways for abusing privileges, abuse result (whether caused a harm to the system, actors or stakeholders of the system) etc. in order to build effective abuse case models for a certain system. Description of each abuse case should be complete to know, such as, how a malicious agent could gain an unauthorized control of the system or subpart of the system. Abusing privileges is one of the main security threats to the system [17], thus it is very important to describe the privilege(s) for each actor in the system as well as the different ways to abuse these privileges. Regarding the Mailing System the User privileges are limited and they include the ability to

- Changing account password.
- Sending and receiving emails.
- Adding and deleting email addresses.

Thus, changing account password by the User himself might endanger the User account in case the chosen password is easy to guess. Finally, it is highly recommended that each abuse case should include short description of the specific harm that will occur as a result of the abuse [17].

In order to build an abuse case model, requirements engineer should study and investigate the use case model of the same system to figure out the corresponding component (malicious components or actors) for the abuse case model. McDermott et al. [17] provide a set of steps as a process that helps in building an abuse case model for a system. The following is the description of these steps:

1. ***Identify the abuse actors***. After identifying the actors of the use case model, actors for corresponding abuse case model should be identified. Basically, if an actor in the use case model might act maliciously or attempt to harm the system, then a corresponding actor in the abuse case model should be added (e.g. from Figure 5, if one student may attempt to spread some viruses via the Mailing System, then an actor such as *"BAD STUDENT"* should be added to the corresponding abuse case model as shown in Figure 6). This does not mean that only the corresponding actors will be created for the abuse case model, also new actors such as hackers might be added to the abuse case model. The actors in abuse case model should be descried in detail; therefore, McDermott et al. [17] suggest three characteristics that should be descried clearly for each actor in the abuse case model. These characteristics are: the actor resources, skills and objectives. We think that these characteristics are highly related and dependants on each other; the objectives depend on the available recourses as well as the actor skills and the environment of the attacked system as well. In addition, if the

actor objective is to control the whole system, then more than one abuse case description are needed to cover that objective. Therefore, the objective of the malicious actors should be descried clearly in a simple way; otherwise in case of a large objective it should be divided into a set of abuse case s that aims to complete that objective. McDermott et al. [17] discuss the actor objective in the same sense where they described the actor's objectives as long-term goals that the actor seeks to reach using more than one abuse case. Regarding the abuse case model of the Mailing System (see Figure 6) we introduce two actors: *DARK* and *DARTH* with different resources, skills and objectives. Thereby, in order to describe these characteristics for DARK and DARTH in a clear way, we developed a simple table and fill it for both of them as shown in Table 1 and Table 2.

Item	***Description***
Actor name	*DARK*
Abuse cases involvement	• Steal username and password • Capture sent mail • Spread viruses
Actor description: Resources, skills and objectives.	***Resources:*** hardware, software and Internet access to the system. Also, he/she has some of the documentation of the system that he/she intends to abuse; thus he/she might be able to test or simulate his/her intended exploit [17]. ***Skills:*** DARK has good technical skills; therefore he/she knows how to perform a well engineered attack. He/she has good backgrounds about Internet protocols and mailing systems. In addition, he/she can develop the necessary software to accomplish the exploit. ***Objectives:*** his/her objective includes making system unavailable, theft and increasing his/her own skills. ***(DARK might be the ex-administrator for the Mailing System)***
Assumptions	DARK is willing to spend about four months trying to defeat the system security.

Table 1: Description of the *"DARK"* actor

Item	***Description***
Actor name	*DARTH*
Abuse cases involvement	• Steal username and password • Monitor sent mail
Actor description: Resources, skills and objectives.	***Resources:*** hardware (e.g. his/her PC), exploit software (e.g. Scripts Kiddies prepared by others) and Internet access to the system. ***Skills:*** DARTH has limited technical skills; therefore he/she use attack scripts prepared by others as well as some developed tools such as Dsniff [39]. ***Objectives:*** learning how to use the exploit software and trying to demonstrate his/her technical skills.
Assumptions	DARTH is willing to spend about two days trying to defeat the system security.

Table 2: Description of the *"DARTH"* actor

2. ***Identify the abuse cases.*** For each actor defined in step 1, requirements engineer has to determine the interactions of that actor with the system. Therefore, a set of abuse cases should be identified in an abuse case diagram (see Figure 6 that shows the abuse case diagram for the Mailing System). The way of building the abuse case diagrams is as same to the one used in building the use case diagrams; thus the same language

(UML) and symbols are used without any modifications. As a result, each abuse case diagram will contain malicious actors, associations and misuse cases. Label or name for each malicious actor in the abuse case diagram should be different from the name that is used in the use case diagram since duplicate actor name in both use case and abuse case models will confuse the reader [17]. For example, if we intend to define a malicious user for the Mailing System, then the suggested name should not be "*User*" because this name is used in the use case diagram; thereby any other relevant name such as "*Crook*" should be used. In addition, label or name for each abuse case should be related to the contents (e.g. steal username and password). This step aims only to identify the abuse cases without describing them; in other words it aims only to structure the abuse case diagram in high level.

3. ***Check granularity.*** This step aims to provide a highlight to help to decide the number of abuse cases that should be constructed. Generally, deciding the number of abuse cases is a matter of experience and consideration of the developed system [17]. Constructing large number of abuse cases that are unlikely to occur will not provide any help; also constructing few numbers of abuse cases might endanger the system. Therefore, we must be vigilant when deciding to include or discard an abuse case. *"A good abuse case model will have uniform granularity of detail in its cases, and not too many of them"* [17]. In order to decide the number of abuse cases that should be constructed, we suggest the use of some criteria such as (1) the result of the abuse case (amount of harm to the system), (2) possibility of occurrence, (3) whether the abuse case is redundant, (4) estimated development time etc. In addition, we suggest that the abuse case should be constructed in a general form for some situations in order to minimize the redundancy. One example to clarify this is that requirements engineer may define an abuse case such as *"deleting some information from user profile"* or *"modifying user profile"*; the previous two abuse cases might be considered as redundant from security point of view since both of them are related to the confidentiality. Therefore, the abuse case *"modifying user profile"* is defined in general and covers the other one which might be discarded.
4. ***Define abuse cases.*** Defining and describing abuse cases are difficult and tricky task since the aim is to produce good enough description for the abuse cases that have been identified in the previous steps. The description of an abuse case can be carried out using the same way applied with the use case. However, sometimes different ways are applied and this depends on the characteristics of the abuse cases under investigation. Generally, the description of a use case is concerned with a single or sequence of desired transaction; while in the abuse cases it is not clear where the flaws will occur [17]. Therefore, abuse cases describe a family or set of undesirable interactions (malicious interactions) [17]. In addition, the Abuse Cases technique aims to reduce the number of exploits in the system by decreasing the number of overlooked requirements and design flaws. Therefore, description of the abuse cases will include different components, agents and subsystems involved in the abuse cases as well as different paths (alternative ways) of the transactions that may occur to accomplish the same abuse case [17]. In order to accomplish that, McDermott et al. [17] suggest and use a tree or sub-tree of an attack tree to describe different paths of the transactions that might be used to accomplish the same abuse case. The root of the tree represents the system which is under modeling; the leaves represent the resources or components of the system that are the targets of the abuse case and the interior nodes represent subsystems, applications and individual classes within the applications [17]. Finally and according to the used tree, the description of the abuse cases depends on the system interface and its components. Thus, abuse case descriptions cannot be finished until there is enough system structure to work with. In other words; *"as the interface to the system becomes more refined and the specific components are identified, the abuse case can be described"* [17].
5. ***Check completeness and minimality*** **.**This is the last step in the process and it aims to check the description for each abuse case to check whether it describes an interaction

that results in harm to the system user or a stakeholder [17]. Users, security experts and requirements engineer should be involved to ensure that no critical abuse cases have been overlooked. Moreover, checking redundancy, missing data, unlikely abuse cases etc. should be carried out in this step.

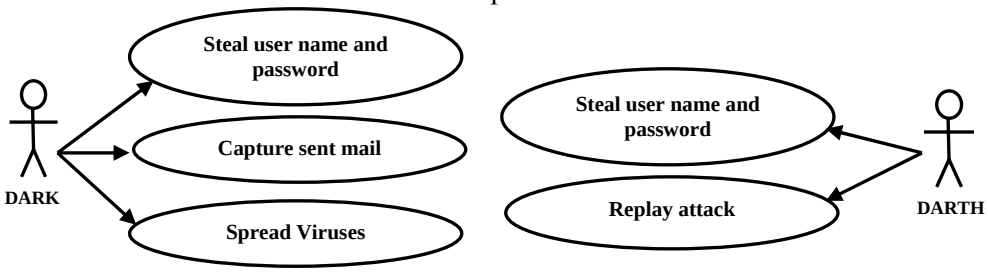


Figure 6: Abuse case diagram for a school mailing system

Referring to the use case model of the Mailing System (see Figure 5); Figure 6 shows the corresponding abuse case model for the same system. We can note from Figure 6 that the abuse case *"stealing username and password"* is defined twice but it differs in the characteristics of the actors (resources, skills, and objectives). The following is the description of the abuse case *"stealing username and password"* for both DARTH and DARK.

DARTH is hypothetically assumed that he/she knows a specific user ID or he/she guesses a valid user ID (user ID is built according to a specific and known algorithm to DARTH). Thus, he/she will use packages and tools that allow him to perform the dictionary attack from a graphical user interface application. For each failure the system displays an error message and asks to try again. The number of possible trials depends on the system design which could be either unlimited or the system will lock the account after three failed trials. If it is limited to a certain number it will minimize the possibility of the attack. Otherwise, DARTH will continue performing the attack until he/she finds the user password. Finally, if DARTH gets the user password he/she can read the user private mails and take a copy of his/her private address book. In the worst case, DARTH might change the account password; as a result the legitimate user will lose his/her private mail account.

DARK will perform a well-engineered attack designed specifically to steal the user ID and password. Thus, he/she might perform an email spoofing attack by sending the user an email which seems to be sent by the administrator. In the content of the message, DARK may ask user to perform specific action such as sending his/her old and new user ID and password for some security reasons (e.g. new password should include alphabet characters, symbols and numbers). Otherwise, DARK may ask the user to install the attached file in order to get access to some new services; in this case attached file might be a program to steal the user ID and password when the user types them (e.g. keyboard sniffer program). If the user performed any of DARK's requests, DARK would get the user ID and password. The same harm that occurred by DARTH attack is expected to occur in DARK attack.

The previous process for building abuse case model seems to be sequential steps; but from our experience, dealing with the system requirements as well as the discussed issues for the requirements engineering process in Chapter 4, requires an iterative process. The main reason behind that is the requirements usually change during the development process. Therefore, a waterfall process might be considered as insufficient for constructing the abuse case model. Thereby, using the mentioned process should be done iteratively.

Regarding the previously described process, we think that checking granularity and checking completeness and minimality steps should be overlapped with identifying abuse actors, identifying abuse cases and defining abuse cases steps since all of these steps are highly related to each other. Defining abuse cases (step number 4) depends on the abuse cases that are identified in step number 2, which also depends on the identified actors in the first step. Therefore, checking granularity and checking completeness and minimality steps should be linked all the time to steps number 1, 2 and 4 in the process. After studying the

whole process, we found that it needs some kind of rearrangement or rebuilding. Therefore, we rearrange and rebuild it as shown in Figure 7. Our vision used to rebuild the process is similar to the requirements engineering process described in Chapter 4. The description of the steps is still the same as mentioned by McDermott et al. [17]. In addition, the comments mentioned during the previous discussion of these steps should be taken into consideration such as, the suggested criteria to decide the number of abuse cases that should be constructed as well as the use of the table to describe the defined abuser actors.

Documentation and management is a new step that should be carried out with all other steps. Documentation aims to document and specify the used and discarded abuse cases as well as the reasons behind this decision. Management aims to manage the constructed abuse cases using a database which makes it easer to refer to some abuse cases for update, reuse etc.

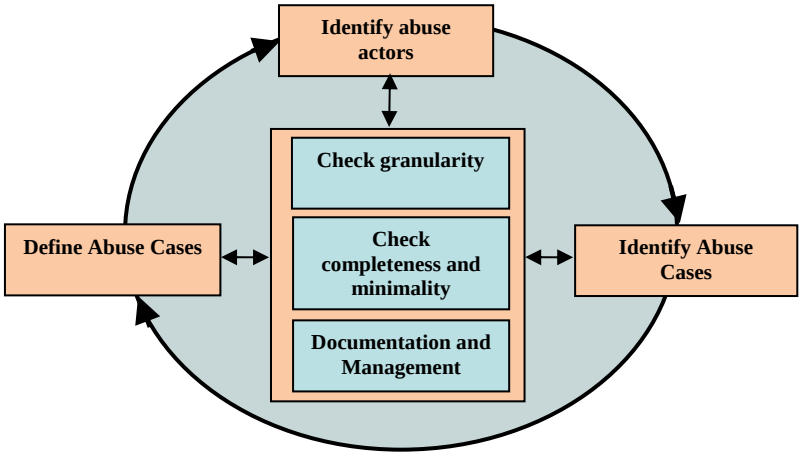


Figure 7: Process for constructing abuse case model

5.1.2.2 ADVANTAGES VS. DISADVANTAGES

Two documents will be produced, one to present use case models and the other to present abuse case models. Essentially, both documents will include UML diagrams as well as descriptions for either use cases or abuse cases. Part of the documentation for abuse cases will be conducted using UML without any modifications; this allows modeling abuse cases in standard notations as well as using available design tools such as Rational Rose [17]. Furthermore, it will be easy for the requirements engineer, stakeholders and developers to understand abuse case diagrams and the used notations. On the other hand, there is no standard for the description of the UML diagrams, the description may include more or little details depending on the considerations of the developed system, requirements engineers experience [17], the situation, budget, project time etc. One opportunity to overcome this is by structuring standard or general worksheets to describe abuse cases in a minimum level.

One of the main disadvantages for using the Abuse Cases techniques is that the description of abuse cases depends on the system interface and its components as mentioned in step 4 in the previously mentioned process. This challenge seems to be hard to overcome and it might require a high level of collaboration between different stakeholders involved in the development process. However, using prototypes during the elicitation phase might provide better overview of the interface and its components for the system under consideration.

Since abuse case models are easier to understand, comparing for instance to the diagrams produced by using the Strategic Modeling technique, they could be used to increase the system stakeholders' awareness and understanding of the security features and system vulnerability. In addition, it is a helpful technique for eliciting, modeling and analyzing security requirements. Abuse case model shows different ways for abusing the system, thus achieving better understanding for the security threats in the analysis phase. However,

defining or finding out all different ways for abusing the system is a matter of experience, creativity and imagination for the creator of abuse cases. In other words, the creator of abuse cases should think and act as an attacker during building abuse cases. Thus and especially for sensitive security software we recommend the involvement of the security experts during the process of creating abuse case models as well as the process of writing abuse case descriptions.

In addition, documentation might be considered as disadvantage for using the Abuse Cases technique and this depends on the size of the produced documentation. Thus, producing abuse case models in a separate document may generate a huge document that might be difficult to be managed, also it may lead to confuse the reader because both use case diagram and abuse case diagram use the same kind of symbols. Consequently, reading such huge documents should be done carefully; otherwise the reader may mix use cases with abuse cases. Using a database or at least Microsoft Excel worksheets to manage abuse cases documents is recommended to overcome the updates, referencing, documents size problems etc.

We believe that one of the biggest challenges for using the Abuse Cases technique is deciding the number of abuse cases that should be developed within the system. Overcoming this problem should be carried out by negotiating abuse cases with different stakeholders further to study the result of each abuse case, possibility of occurrence etc. Another main challenge for using the Abuse Cases technique is the creativity of the creator of abuse case models which depends on his/her imagination, experience, knowledge about security theories, knowledge about different kinds of threat etc. In addition, using assumptions is considered as another main risk for using the Abuse Cases technique (e.g. McDermott et al. [17] assume that Script Kiddie is willing to spend about 24 hours trying to defeat the security of a particular system). Therefore, any assumption should be discussed very carefully by requirements analysis and security experts and it is preferable to be discussed before using that assumption; this helps in designing and using a realistic assumption. On the other hand, serious attackers will try to undermine the used assumptions for building the system. For example, if the system design assumes that connections from web server to the database server are always valid, then an attacker will try to make web server send inappropriate requests to access valuable data [38]. Informed brainstorming might be used to overcome and guide both the creation process of abuse case model as well as the assumptions; thus requirements engineer, requirements analysis and security experts should ask many questions about the system design to identify the likely weakness places that the system might have [38]. In the same sense covering all kind of assumptions and available attacking tools and software is a very hard task; thus the most realistic and likely one should be covered according to the previously mentioned issues.

Using this technique helps in covering security concepts (CIA); Confidentiality can be covered by designing an abuse case for the disclosure of the private information (e.g. capture sent mail). Integrity can be covered by designing an abuse case for the possible ways for modifying information. For Availability, abuse case should describe how a malicious actor may attempt to break the system availability by deleting some files or using DOS attack. Building abuse case diagrams, after constructing the corresponding use case diagrams, helps in accelerating the performance of finding, understanding and building abuse case diagrams and their descriptions. The Abuse Cases technique can be used in practice in a high level since it is built upon the Use Cases technique which is considered as a more successful technique than textual requirements in capturing stakeholders needs [18]. However, this does not mean that the requirements specification (textual requirements) is ineffective or the Use Cases and Abuse Cases techniques should replace the requirements specification; instead, requirements engineer should use both of them with concentration on the most suitable technique for capturing and modeling security requirements.

5.1.2.3 STAKEHOLDER INVOLVEMENT

There are many types of stakeholders that might be involved in order to use the Abuse Cases technique. We believe that the following are some of them [25] [30] [33] [38]:

- Requirements Engineer: Responsible for eliciting, documenting, negotiating and checking the system requirements. Also and since he/she is responsible for building abuse case models, he/she should collaborate with the system analyst to check the competence of these abuse case models.
- End-user: Usually interacts with the system; and is responsible for providing use cases as well as some of the abuse cases with the new system and his/her expectations about it. Use cases will be used by requirements engineers to create the corresponding abuse cases.
- Project Manager: responsible for assessing the feasibility of developing the abuse cases as well as the system in general. Also, he/she is responsible for planning, monitoring, and controlling the project and guiding the software development team to the successful delivery of the software.
- Security Expert: Responsible for providing solutions and alternatives to overcome the security threats and vulnerabilities within abuse cases. In addition, they are responsible for checking the assumptions and identifying the places where the system is likely to have weaknesses.
- Requirements Analyst: Responsible for studying, analyzing, assessing and finding conflicts between abuse cases. In addition, they should cooperate with security experts in order to understand how the system features might be attacked and how to protect software as well as checking the assumptions.
- Developers: Responsible for implementing security requirements. Also, they may provide new abuse cases according to their experience.

5.1.3 MISUSE CASES TECHNIQUE

Misuse Cases technique is consider as an interesting technique since it offers the possibility to look at functional requirements (represented by use case models) and security requirements (represented by *misuse case models*) together in a simple way. Misuse case models, their concepts and some of their features are derived and inherited from the use case model. Sindre et al. [18] developed this technique because using Use Cases technique is not adequate to describe security requirements. Thus, they make some modifications to the UML notation used in the use case diagram to capture and model security requirements. Also, they defined some additional terminologies such as *mis-actor*, *prevent* and *detect*.

Misuse case is defined as a "*special kind of use case, describing behavior that the system/entity owner does not want to occur*" [18]. Generally, the Abuse Cases and Misuse Cases techniques have the same purpose with some similarity as well as some differences in the internal process and documentation. A comparison between the Abuse Cases and Misuse Cases techniques can be found in section 5.1.4.

5.1.3.1 MISUSE CASE MODELING PROCESS

Any malicious agent who may misuse the system is named as "*a mis-actor*". Accordingly, *a mis-actor* is defined as "*a special kind of actor who initiates misuse cases*" [18]. *Misuse case model* is highly concerned with representing misuse cases and mis-actors with the corresponding use case models in the same diagrams without confusing the reader (see Figure 8). Achieving that without modifying UML notations to distinguish between misuse cases, mis-actors, use cases and legitimate actors is consider as inadequate, especially if the diagram represents a huge system. Thus, Sindre et al. [18] confirmed that it is *"unwise to mingle use and misuse in one diagram without making it absolutely obvious which is which"*. As a result, the misuse cases and their actors are portrayed in an inverted format comparing to the use case diagram as shown in Figure 1. Fortunately, the developed diagram

(e.g. Figure 1) includes both use cases and misuse cases together in a simple and obvious way.

In order to make misuse case model dynamic, the standard *"includes"* and *"extends"* relations from UML are used. Also, new kinds of relations were added by Sindre et al. [18] which are *"detects"* and *"prevents"*. All of these relations usually do not have a name; instead the relation name (e.g. includes) should appear beside the connector line as shown in Figure 8. The meaning of these relations is as follows:

- ***Includes relationship***, an arrow is drawn from a use case X to a use case Y to indicate that X includes the functionality of Y. This relation supports the reuse of functionality of another use case.
- ***Extends relationship***, an arrow is drawn from a use case X to a use case Y to indicate that X is a special case behavior of Y which is a more general process (X is a subset of Y). In other words, it is a generalization relationship where one use case extends another use case by adding some actions to it.
- ***Prevent relationship*** an arrow is drawn from a use case X to a misuse case Y to indicate that the function provided by X prevents the activation of Y, at least in some cases [18].
- ***Detects relationship,*** an arrow is drawn from a use case X to a misuse case Y to indicate that the function provided by X detects the activation of Y, at least in some cases [18].

The "*at least in some cases*" part in both prevent and detects relationships is important to notice since it is very difficult to detect or prevent all possible ways of performing a certain misuse case using only one countermeasure (an action taken to ensure software security) technique [18]. For example, using strong encryption algorithm (countermeasure) such as RC5 or blowfish can prevent an attacker from getting username and password easily but this depends on the algorithm key length, block size etc.

In our school Mailing System example, we did not cover all actors, security threats, use cases and misuse cases instead we tried to keep it simple to clarify the Misuses Case technique. Thus, in Figure 8 we use only one mis-actor that is introduced previously and used in describing the Abuse Cases technique, named *"DARK"*, who may perform a set of misuse cases such as "*stealing username and password*". The figure includes four kinds of relationships; the relationships *"Includes"* and *"Extends"* are used to connect the use cases to each other. However, the relationships *"Includes"* and *"Extends"* can be used to connect the misuse cases to each other [18]. On the other hand, the relationships *"Prevent"* and *"Detect"* are used between the use cases and misuse cases.

In order to build a misuse case diagram, Sindre et al. [18] developed an iterative process that consists of five steps for that purpose. The description of these steps is as follows:

1. Build use case diagram using UML notations. Thus, the focus in this step will be on the legitimate actors and their use cases. As a result, Figure 5 is considered as a subpart of a real mailing system which may include other legitimate actors such as administrator, server, teachers etc.
2. Introduce main mis-actors and misuse cases including most reasonably likely threats. A number of mis-actors depend on the system under development and it may include external mis-actors such as competitors [18] or hackers; also it may include internal actors such as mis-user who acts maliciously.
3. Explore potential relations between misuse cases and use cases. This step is essential because some threats can be widely achieved by using the normal functionalities of the system [18], for instance one kind of DOS attack can be performed by flooding the system with user-registration functionality.
4. Update diagrams by introducing new use cases in order to detect or prevent the defined misuse cases. This step aims to complete the diagram; thus the diagram should

include all kinds of relationships between misuse cases itself, use cases itself and between misuse cases and use cases.

5. Continue with more detailed requirements documentation.

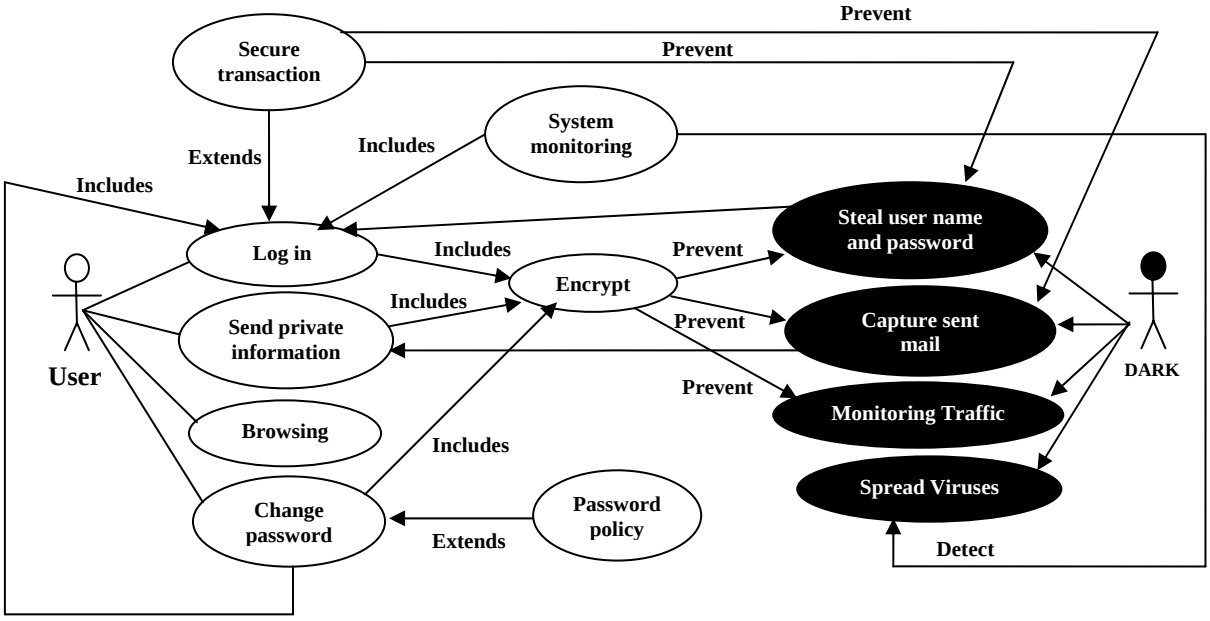


Figure 8: Use and misuse cases for a school mailing system

This process is an iterative process and the sequence of the steps seems to be reasonable. In step 1 requirements engineer needs to sketch the use case diagram for the system under consideration. Steps 2 and 3 are concerned with building misuse cases and link them to the use cases. Step 4 aims to find some use cases that might be not requested by system owners but they are helpful to detect or prevent the misuse cases. Finally, step 5 indicates that there is more work to be done after modeling the use cases and misuse cases [18] such as documentation and management for the use and misuse case models. On the other hand, Sindre et al. [18] recognize that the process for building misuse case diagram still incomplete since there are many considerations which cannot be expressed from misuse diagrams; thus they leave the door open for other researchers in order to complete the process, therefore they mention some of the missed considerations like:

- Detailed descriptions about mis-actors including their motivation, knowledge and destructive capability. This might be overcome by using the three characteristics proposed by McDermott et al. [17] to describe clearly each mis-actor in the misuse cases.
- Likelihood of various threats as well as the expected harm to the system and stakeholders. This might be overcome by adding short description for the expected harm.
- Other possible countermeasures of security threats. One suggestion to avoid this miss consideration is that all kind of countermeasures should be discussed carefully involving security experts and requirements engineer, after that the chosen one should be used in the diagram.
- To what extent the countermeasures can be helpful and affective to detect and prevent security threats. Overcoming this issue depends on the used countermeasures. For the encryption countermeasure the security of encryption algorithm depends on the secrecy of the algorithm key not on the secrecy of the algorithm [40]; thus using large key size helps in decreasing the possibility to decrypt the content of the message.

The possibility for the system owner to have and use the suggested countermeasure in reality might be consider as another missed consideration (e.g. from Figure 8, if there is no any kind of system monitoring are performed by a legitimate actor, then how we can detect

whether there is a process of spreading viruses). Thus, a detailed description for each of the used countermeasures should be provided to make things clear. Also, providing countermeasures alternatives is preferable.

The above mentioned issues regarding the process of building a misuse case model does not mean that the process is unhelpful instead it indicates that the process is incomplete; thus it might be used to perform some parts of the elicitation and modeling. Also, misuse case model is a very helpful technique to increase the system stakeholders' awareness for security threats. Moreover, this technique, as well as Abuse Cases technique, helps in involving the security requirements and threats in the software development process from the beginning.

5.1.3.2 ADVANTAGES VS. DISADVANTAGES

Comparing misuse case model to the abuse case model, only one document will be produced to present the use cases and misuse cases models. Essentially, the documents will include UML diagrams as well as descriptions for both use and misuse cases. Producing only one document can be considered as an advantage or disadvantage and that depends on the situation (i.e. if the investigated system is a huge system, then the produced document will be too huge, thus it will not be easy to manage and understand. On the other hand, if the investigated system is small, then the produced document will be easy to manage and understand). Each diagram will hold use cases, misuse cases and the relations between them. Generally, it will be easier for the requirements engineer, stakeholders and developers to understand the misuse case diagram. Moreover, presenting use cases and misuse cases together is very important in terms of:

- Showing relations between use cases and misuse cases where some of the *"Intended sabotage may often be achieved by means of ordinary functions that the system has to support"* [18].
- Clarifying which actors will be affected by a certain misuse case (e.g. user will be negatively affected if his/her password is stolen). These benefits will be impossible to be achieved easily if totally separated diagrams were used [18].

On the other hand, especially for a complex system such as bank system, combining misuse cases and use cases and different kinds of the relations may lead to increase the complexity of the produced diagrams. Thus, building misuse and use case diagrams should be done carefully. Otherwise, a complex diagrams will be produced which make it difficult to be understood by the reader or the requirements engineer; thus failing in reaching simplicity as it is supposed. Also, misuse case descriptions have the same problems that are mentioned previously for the abuse case descriptions and it might be overcome using the same way.

As it seems to us, the biggest challenges for using the Misuse Cases technique exists in step 4 in the previously mentioned process; precisely introducing additional use cases (e.g. from Figure 8 *"Encrypt"* or *"System monitoring"*) and their relationships with the other use cases and misuse cases. According to the process, this step seems to be essential but on the other hand it indicates that the used countermeasure(s) covers the security threats which might not be true in all cases. For example, using an encryption algorithm to encrypt the username and password during the login may help to prevent stealing username and password but this depends on the encryption algorithm and its key length; thus an attacker may control and utilize a cluster of computers in order to break and decrypt the content of the message; therefore getting the password. Accordingly, introducing additional use cases can be discussed from two points of view; the first one is the additional use cases only provide one possibility to cover one security threat but this does not mean it is the only way or the best way to overcome the security threat (e.g. system monitoring is one way to detect DOS attack, another way is by make some limitation for number of the request that can be sent from the same source at some specific time). This overview might be very helpful in the analysis phase as well as design phase since it open the door for the requirements analyst and system designer to think about different techniques that might be used to overcome the

security threat and choose the most significant one. The second point of view is derived from the principle of the requirements engineering process; where we can note that the requirements engineering process aims to find out system requirements without deciding how to implement them since it is the duties of the design phase. Thus, using countermeasures cannot be considered as a weak-point for the process. Anyhow, informed brainstorming might be used to help to decide which countermeasure technique is the most suitable for a certain case.

The Misuse Cases technique uses assumptions like the ones used in the Abuse Cases technique in order to describe the misuse cases, thus the previously mentioned issues to the Abuse Cases technique in section 5.1.2.2 should be taken into the consideration. By using the same way that has been used to discuss the Abuse Cases technique, we can note that using the Misuse Cases technique helps in covering security concepts (CIA).

5.1.3.3 STAKEHOLDER INVOLVEMENT

There are many types of stakeholders that might be involved in order to use the Misuse Cases technique. Those stakeholders are the same as those mentioned for the Abuse Cases technique (see section 5.1.2.3).

5.1.4 DISCUSSION OF THE ABUSE CASES AND MISUSE CASES TECHNIQUES

This section holds a discussion about the similarity and differences between the Abuse Cases and the Misuse Cases techniques, possibility for combining both techniques together and some suggested templates for documenting the abuse cases and misuse cases.

5.1.4.1 ABUSE CASES TECHNIQUE VS. MISUSE CASES TECHNIQUE

The similarities between the Abuse Cases and Misuse Cases techniques include but are not limited to the sharing of the UML notations, both of them are derived from the Use Cases technique, similarity in the processes etc. These similarities can be noted from the previous sections 5.1.2 and 5.1.3. Thus, in this section we tried to focus on the difference between both of them.

Referring to the abuse case definition, we can note that misuse cases and abuse cases are not exactly the same. As it is mentioned earlier, according to the Abuse Cases technique, the case will be classified as an abuse case if it results in actual harm to the system, actor(s) or stakeholder(s). On the other hand and according to the Misuse Cases technique, the case will be classified as a misuse case if the system owner does not want that case to occur. One example to clarify that is the misuse case *"monitoring traffic"* (see Figure 8) which is considered as a passive attack. A passive attack attempts to learn or make use of information from the system but does not affect system resources [40]. Thus, suppose that the content of the messages are encrypted using strong encryption algorithm, *"DARK"* as an attacker might still be able to observe the pattern of these messages. Consequently, even if it is not possible for *"DARK"* to decrypt these messages, he/she may perform an analysis process on the traffic in order to guess the nature of both the messages and the communication. Passive attacks are very difficult to detect since they do not involve any data alteration [40]. The previous example might be considered as a misuse case since system owner does not want such cases to be occurred. However, *"monitoring traffic"* might not be considered as an abuse case since it does not cause an actual harm to the system, actor(s) or stakeholder(s). Nevertheless, if the misuse case *"monitoring traffic"* is updated to include the offline dictionary attack or used to determine the location and the identity of the communication hosts then it might be considered as an abuse case since it will affect the confidentiality thus an actual harm to the system or actor(s) might occur.

One of the main notable differences between Abuse Cases technique and Misuse Cases technique is that the misuse case models within the Misuse Cases techniques explore the relationships between use cases and misuse cases. Clearly and by looking to the Figure 6 and

Figure 8 we can note the additional relationships that have been added to the misuse case model to link the use cases and misuse cases together. Another notable difference is the way of documenting and presenting use case diagrams and abuse case diagrams on one hand and the way of documenting and presenting use case diagrams and misuse case diagrams on the other hand. Obviously, we can note that in the Abuse Cases technique the abuse cases are represented in a separate diagram and described in a separate documentation and the same thing has been done to the use cases; while in the Misuse Cases technique, both use cases and misuse cases are represented in the same diagram and described in the same documentation.

As a result, it is clear from the previously mentioned sections that the Abuse Cases and the Misuse Cases techniques have the same purpose and the similarities regarding defining abuse cases or misuse cases are more than the differences. Finally, deciding which technique (Misuse Cases technique or Abuse Cases technique) is better is not easy as it might seem, since each of them has its advantage and disadvantage as discussed in sections 5.1.2.2 and 5.1.3.2. However, we think that Misuse Cases technique is more organized and helpful than Abuse Cases technique, because it represents both misuse cases and use cases in the same diagram that shows the relationship associations between them which are very important to provide an overview about the relationships between use cases and misuse cases (i.e. it would be impossible to use the *"includes"* relationship associations between misuse cases and use cases if we totally separate the misuse and use case diagrams [18]).

5.1.4.2 COMBINING ABUSE CASES TECHNIQUE AND MISUSE CASES TECHNIQUE

Combining the Abuse Cases and Misuse Cases techniques together is very helpful in order to overcome some of the weak-points. Thus, we suggest using the process of the Abuse Cases technique to produce diagrams like the one in the Misuse Cases technique. By combining the processes of both the Misuse Cases and Abuse Cases techniques we can inherit some benefits that exist in one process but not in the other such as:

- Describing malicious actors is missed in the misuse cases process while in the abuse cases process it is carried out in a good way.
- Building diagrams using the misuse cases process for the reason that is previously mentioned.

Thus the final iterative process will be as follows:

1. Build use case diagram.
2. Identify mis-actors.
3. Identify misuse cases.
4. Check granularity to decide approximately the number of misuse cases that should be defined.
5. Define misuse case.
6. Explore potential relations between use and misuse cases.
7. Update diagram by introducing new cases in order to detect or prevent misuse cases.
8. Check competence and minimality.
9. Documentation and management

The descriptions of these steps are already mentioned and they should be used iteratively. Step 9 should be carried out with all other steps (i.e. after performing any change, documentation should be carried out).

5.1.4.3 ABUSE CASE AND MISUSE CASE DOCUMENTATIONS TEMPLATES

Another issue regarding both the Abuse Cases and Misuse Cases techniques is both of them are considered as informal techniques. Thus, standard templates for documenting abuse cases and misuse cases are massively needed. Accordingly, Sindre et al. [41] suggest a template for describing misuse cases built upon some of the proposed templates for use cases by [42] [43]. The suggested template to describe misuse cases, can be used to describe abuse cases as well, includes many items such as [41]:

- Misuse case Name: intuitive name that uniquely identifies the misuse case.
- Summary: description of interaction in one or two sentences.
- Author: creator for misuse case.
- Date: creation date.
- Basic path: the most common path to accomplish the goal of this misuse case.
- Alternative paths: other different ways to accomplish this misuse case.
- Capture points: countermeasure that might be used to prevent and/or detect the misuse case at particular steps.
- Extension points: Optional paths. *"These will be options taken by the mis-user, for instance to hack around hindrances, whereas capture points work against the mis-user"* [41].
- Triggers: The entry criteria for the misuse case, i.e., what initiates it.
- Preconditions: Conditions that must be true before the misuse case can be performed.
- Assumptions: Conditions assumed true to perform misuse case. *"But contrary to preconditions, assumptions cannot be guaranteed by the system itself"* [28].
- Post-conditions: description of what will be true when the misuse case is completed.
- Related business rules: description of the system boundary in an operational manner. This can be coupled to more declarative business rules.
- Potential mis-user profile: description of the malicious user who initiates the misuse case including his/her skills, used tools etc.

We suggest some other relevant items to the misuse template which are expected harm, comments and violated security concept. In the expected harm item, the requirements engineer and/or analyst can describe the expect harm caused by the misuse case; this might be used to assign a number that denotes the risk of the misuse case. The comments may include information about the attack or defense; these comments can be written in general such as the thoughts of the requirements engineer regarding this attack and his/her suggestions for the countermeasures that might be used to overcome this attack; these comments may be considered as a helpful comments to the analyst during the analysis phase. Finally, in the violated security concept item, the requirements engineer can mention one or more of the security concepts (CIA) that will be affected by accomplishing the misuse case; this helps in deciding the suitable countermeasures that might be used to overcome the attack.

5.2 DATA SENSITIVITY AND THREAT ANALYSES TECHNIQUE

The main reason behind developing the Data Sensitivity and Threat Analyses technique comes from and is built upon the software architect's point of view where they think it is often hard to understand and it is too general to directly apply security methodologies that have been developed by security specialist [5]. Thus, failure in understanding security theories, threats, vulnerabilities etc. will end up with inadequate secure application. *Data sensitivity analysis* is concerned with describing the business impact if the data is disclosed or altered by an unauthorized person as well as the business impact if the data is unavailable to the legitimate user. On the other hand, *Threat Analysis* is concerned with a likelihood of somebody attacking the system under consideration [5].

Therefore, the Data Sensitivity and Threat Analyses technique offer a means to focus on the value of the data that needs protection as well as whether an attacker has an interest in disclosing the data processed by the application. As a result, and taking into consideration time to market constraint, using this technique helps in saving project time and effort from protecting the insignificant data; thus provides strong enough protection for the most significant data within the system and develop adequate security application as a result.

5.2.1 DATA SENSITIVITY AND THREAT ANALYSES PROCESSES

There is no formal approach for Data Sensitivity and Threat Analyses technique. Hence, effectiveness in using this technique is a matter of experience as well as the consideration and understanding the value of the data and information within the developed system. However, this does not mean this technique is inadequate instead it indicates that using this technique should be done carefully by mainly involving system stakeholders, since they highly recognize the most important data and information within the system.

In order to provide sufficient details and explanation for the focus of this technique, Kis [5] mentioned two commonly observed cases – antipatterns. In the first case, a well-known (perimeter security) model is applied in a new context without analysis of the security requirements. In the second case, the impact of lacking data sensitivity classification and threat analyses is considered. Therefore, two main problems faced daily in practice are investigated. First, building secure application without spending extreme time and effort by applying some known solutions like using simple password authentication. Applying some known solutions might seem to be a reasonable idea. However, applying such solutions without a thorough understanding of security requirements does not provide adequate protection within the specific context. The second issue is designing an application that fails in understanding the real value of data we need to protect (without performing the data sensitivity analysis and analyzing if an attacker has an interest to compromise the data processed by the application). Consequently, without applying the Data Sensitivity and Threat Analyses technique, the application security requirements cannot be properly defined, and the solution will not provide an adequate security [5].

We discussed the most important issues regarding the Data Sensitivity and Threat Analyses technique using a school registration system as a supporting example for more clarification. The school registration system used to store the data and information for each student including his/her personal information, registered courses, completed course, grades etc. The system is accessible by student through Internet where they can perform some functionality such as registration for new course or exam, cancel registration, update personal information, checking courses grades etc.

Thus, we use the same elements used by Kis [5] to present our antipattern and explain the Data Sensitivity and Threat Analyses technique. Therefore, first we present the description of a problem and some relevant background information. Next we analyze the context in which that problem usually occurs and faulty beliefs that lead to an antipattern solution. Then the antipattern solution is presented and its security impact is analyzed to provide a solution for the initial problem. Finally, the helpful symptoms for diagnosing the antipattern are presented.

Before starting discussing the school registration system it is worth to mention that presenting antipattern seems to be done in the opposite way. In other words, to understand the benefits of the antipattern we have to imagine the process of developing the system without involving the use of the Data Sensitivity and Threat Analyses technique. Then, the impact of using the technique on the security solution should be mentioned and discussed.

The School Registration System – Antipatterns

Problem: The statement of the problem is pretty simple: we have to secure the school registration system. Based on this problem statement, it is natural to ask some questions such as: is the system under investigation is a typical application; can we simply apply a typical security solution? [5].

Background: Since the system will be accessible by different kind of students via Internet; thus using typical security solutions that have been successfully used with the isolated system (i.e. mainframe computer and limited number of terminals) might be considered as insufficient. Using passwords to control user access to the system is one of the typical security solutions that have been used successfully and considered as adequate before

Internet appearance because *"only a limited number of users and administrators would have physical access to the system terminals"* [5]. However, using Internet changed this concept since unlimited number of legitimate and malicious users may access the system using different terminals from inside or outside the organization. Therefore, without analyzing security requirements we cannot verify whether the provided solution is adequate for the developed application [5].

Context: Information security requirements should be gathered in the elicitation phase of the requirements engineering process. These requirements will be used to design the adequate solution in order to protect the information processed by the school registration system.

Forces: There are two main forces that influence the quality of the proposed security solution which are time to market [3] and difficulty to apply general system's security theory in the software development [5]. Thus, in order to design an effective security solution, designer should have sufficient level of information about the security theories that might be used to secure system. Also, sufficient amount of time is needed in order to develop the security solution. Unfortunately, the education of typical software designer, architect and developer usually covers only very basic security topics [5]. Therefore, additional knowledge about different sorts of security theories is needed to provide an adequate security solution.

Faulty Beliefs: Depending on the circumstances, different kinds of faulty beliefs might occur. Faulty beliefs include but are not limited to the following issues:

- Security can be added after building the system.
- System stakeholders do not know what they need regarding the information security [5].
- All the data within the system have the same sensitivity and should be protected on the same level.
- Encryption will solve all kinds of security problems.

Antipattern Solution (This Solution Based on the Faulty Beliefs): Mostly, without analyzing the data within the system under consideration and security requirements a uniform protection of the resources in the application will be implemented or some security solutions that are perceived as 'strong' (e.g. *"usage of a strong encryption algorithm without real understanding why"* [5]) will be used.

Consequences: The provided solution without applying the Data Sensitivity and Threat Analyses technique may lead to inadequate protection of the resources we have to protect [5]. Therefore, some sensitive data might not be protected enough, while unnecessary effort and money might be spent to protect data that does not need strong protection. Also, we did not measure if an attacker is interested in disclosing the information within the application. Generally, the security solution, such as user authentication, which is based on valid assumptions in the mainframe era (e.g. limited number of users has access to the application), fails to protect an enterprise application from the new threats related to the Internet / Intranet environment [5].

Symptoms: Generally, security requirements specification is postponed until the late phases of the application development [5] which affects architects' decisions to decide the best security solution for the application. Also, requirements engineer mostly declare that customers do not know what they need regarding application requirements as well as security requirements. The statements that might be used by the project team are [5]:

- We will use the latest version of the security product XYZ.
- We will encrypt everything
- 'Why is that mainframe security solution not acceptable while it was fine before?'

Refactored Solution: During this section we used the school registration system example to understand the impact of using the Data Sensitivity and Threat Analyses technique in the

security solution. Since the school registration system used to store the data and information about students, these data and information might be classified in different categories according to the security concepts (CIA). Thus, using data sensitivity analysis, from our point of view, may identify the following data categories: (where 'C' means confidential, 'I' means integrity and 'A' means Availability)

- . Student name, phone number, date of birth, program of study, mailing address, registered courses, completed courses and address (I,A)
- Course grades and social security number (C,I,A)

From the previously mentioned data categories we can note that student name, phone number etc. are publicly known information thus it does not need to be hidden. However, the system should ensure that these data should not be modified by unauthorized user. On the other hand, social security number is very confidential thus it should be hidden and the system should ensure that it cannot be modified by unauthorized user. Generally, the whole system information should be available for the legitimate users. It is obvious that not all the data has to be protected in the same way.

After performing data sensitivity analyses which determine only the damage that would happen in case an attacker gets hold of the data [5], the next step should be performing threat analysis (are there malicious users interested in attacking the application?). Thus, we might think in the following cases:

- It is highly unlikely that somebody would try to alter name, telephone number, date of birth, program of study and current registered courses for a student unless the attacker wants to prevent the victim from getting grades on the courses.
- A malicious user might be interested in getting course grades and social security number for specific student.
- A malicious user might be interested in altering the course grades, completed courses and email/residence address of a specific student. This attack might be performed by a malicious student or an attacker from outside the school.
- It is highly unlikely that disgruntled employee might try to make registration system unavailable.

From Threat Analysis, we can note that course grades, social security number and list of completed courses should be protected in a sufficient way. The school registration system uses to provide an overview about how using the Data Sensitivity and Threat Analyses technique help in understanding the value of each piece of the data within the system and deciding the appropriate way to protect them. Thus, data sensitivity analysis might help in making data properties more clear and highlighting effects of misusing them.

Generally, Kis [5] suggests a solution to bridge the gap between software development theory and practice on one side and general system's security theory on the other by integrating a general system security theory into the existing software development methodologies. However, there is no exact security theory that covers a wide range of vulnerabilities such as technical vulnerabilities of the system (e.g. confidential data is stored in the clear text) and human factor related problems (e.g. weak passwords) [5]. Generating such a general security theory should be carried out from the perspective of the people who designed and implemented the system: software architects and developers.

5.2.2 ADVANTAGES VS. DISADVANTAGES

The main weak-point, from our point of view, of the Data Sensitivity and Threat Analyses technique is that it has informal process. Thus, one suggestion to minimize the effectiveness of the informality is by conducting a formal meeting to discuss and classify the data and information processed by the system. In other words, the discussion and classification for the value of the data should be done in groups consisting of requirements engineers, system stakeholders, architects, security experts and developers. The main advantage should be

given to the system stakeholders since they recognize the value of the data processed by the application as well as the classification of the data as confidential and/or integrity should be carried out according to their consideration. However, all kind of data should be available for the legitimate users. As a result, we recommend using the Data Sensitivity and Threat Analyses technique with other formal techniques that are used for modeling and analyzing security requirements such as Abuse Cases techniques, Misuse Cases techniques etc.

Another disadvantage is that data might be classified in different categories depending on the point of the view of the system stakeholders (e.g. personal information such as phone number or address might be considered as confidential for one student while it considered as non confidential for another student). Also, architects and designers may classify the data in different ways. Thus, conducting some meeting between different stakeholders involved in the development process might help in coming out with a general agreement for classifying the data in specific categories.

As it seems to us, data sensitivity deals with the application security requirements from the architecture and design point of views. Data sensitivity aims to classify the processed data and information by the application into different categories and this cannot be fully performed until we have the initial design of the application.

The main lesson learned from the antipatterns discussed by Kis [5] is, the security solutions that have been used and considered to be suitable for mainframe applications are considered as inadequate for Internet applications. In general, it is not necessarily that the security solution that is considered as being suitable for one application should be suitable for another application.

5.2.3 STAKEHOLDER INVOLVEMENT

There are many types of stakeholders that might be involved in order to use the Data Sensitivity and Threat Analyses technique. Those stakeholders are requirements engineer, users, architects, security experts and developers where the duties of requirements engineer, users, security experts and developers are the same as mentioned in section 5.1.2.3. However, the duties for architects might be as follows [56]:

Architects: Responsible for understanding the business requirements and constraints of the system from the customer or requirements engineer and convert them to technical design. Also, deciding the design and the architecture that should be used to fulfill the system requirements and explain this in detail to the team members in order to reduce waste of time and efforts in the development activity.

5.3 STRATEGIC MODELING TECHNIQUE

Strategic Modeling is a technique concerned with modeling relationships among strategic actors using I* framework. In order to elicit, identify and analyze security requirements, the Strategic Modeling technique offers agent-oriented analysis, goal-oriented analysis and scenario-based analysis which are used to explore security requirements in the system under investigation.

In order to illustrate the Strategic Modeling technique, Liu et al. [6] used the Napster and Gnutella peer to peer (P2P) systems as an example. However, we used Kerberos version 4 as an example to illustrate the technique. In addition, the set of analysis techniques suggested by Liu et al. [6] have been used with the Kerberos example. In the next subsections a general overview of Kerberos and I* framework are provided before discussing the Strategic Modeling technique.

5.3.1 KERBEROS

Kerberos is an authentication service developed as a part of Project Athena at MIT [40]. Kerberos offers a centralized authentication server with main functionality that aims to

authenticate users to the application servers and vise versa. Particularly, Kerberos address a problem of an open distributed environment where users at workstations wish to access services on servers distributed over the network. In such environment, a workstation cannot be trusted to identify its users correctly to network services. Therefore, we would like servers to be able to limit access to authorize users and to be able to authenticate requests for services [40]. In other words, Kerberos addresses existing threats such as a malicious user may pretend to be another user in the workstation in order to gain access to a particular service, a malicious user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation etc.

Kerberos environment contains *Kerberos server*, *a number of clients* and a *number of application servers*; such an environment is referred to as *a realm*. Kerberos server must have the user ID and hashed password of all participating users in its database. Also, Kerberos server must share a secret key with each application server that provides services as well as it must share a secret key with other Kerberos severs in other realm [40]. Each Kerberos server contains an *authentication server (AS)* and *ticket-granting server (TGS)*. AS knows the hashed passwords of all users and stores them in a centralized database. In addition, the AS shares a unique secret key with each application server as well as with the TGS. TGS issues tickets to users who have been authenticated by AS. Ticket that has been generated by TGS is encrypted with a secret key known only to the AS and the TGS [40].

By analyzing the dependency relationships between Kerberos server, client and application server, a set of questions can be answered such as: Who are the potential opponents of the system? How do they threaten the security of the system? Why do they want to attack? What counter-measures can be suggested to the system designers and participants? Which one of them is more effective? [6].

5.3.2 I* FRAMEWORK

Before providing an overview of I* framework, it is worth to mention that I* framework is the modeling language of Tropos [44] [45] and throughout its evolution it has included constructs for the classical concepts of both goal and agent orientation such as *goals*, *actors*, *agents*, *roles*, *tasks*, and *social dependencies* [44].

I* framework has been developed for modeling organizations, and relationships among strategic actors. I* framework supports goal- and agent-oriented strategic modeling and analysis of requirements, with emphasis on non-functional requirements [6] [7]. It consists of two main modeling components which are the Strategic Dependency (SD) model and the Strategic Rationale (SR) model.

5.3.2.1 STRATEGIC DEPENDENCY (SD) MODEL

The SD model is a diagram used to describe the dependency relationships among various actors. Each node represents an actor and each link between two actors indicates that one actor depends on the other to achieve something (social dependencies). Social dependencies are characterized by a *resource*, *task, goal* and *soft-goal* [44] [45] (i.e. actors depend on each other for goals/soft-goals to be achieved, tasks to be performed, and resources to be furnished [6] [48]). By depending on others, an actor may be able to achieve goals that are difficult or impossible to be achieved by himself because the dependency relationship allows an actor to take advantage of capabilities offered by another actor [6] [46]. At the same time, the depender becomes vulnerable if the dependee fails to deliver the dependum since the depender depends on the dependee to achieve its goal [46] [48]. For example, a user can get specific service such as sending an email by login to the server. However, he/she is vulnerable to the service not being provided.

To model complex patterns of social relationships among actors, the SD model differentiates the generic concept of actor (graphically represented as circle) into roles and agents [6] [46] as follows:

- *An agent* is an actor with concrete, physical manifestations, such as a human being or a software/hardware system. Agent can be represented as circle with a line at the top.
- *A role* is an abstract characterization of the behavior of a social actor within some specialized context. A role can be represented as circle with a line at the bottom.

However, using roles and agents (see Figure 9) as special cases of the actor seems to be optional and depends on the software under investigation.

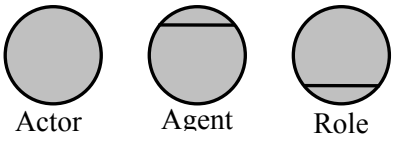


Figure 9: Actors, Roles and Agents

5.3.2.2 STRATEGIC RATIONALE (SR) MODEL

The SR model is a diagram that consists of the actor, intentional elements and the intentional dependency relationships between actors (relationships links). Graphically, an actor may optionally have a boundary containing intentional elements which used to perform some type of analysis as shown in Figure 17. The intentional elements in I* framework are goals, tasks, soft-goals, and resources.

- *A goal* represents the strategic interests of an actor.
- *A task* represents a way of doing something.
- *A soft-goal* represents non-functional requirements and quality concerns of the system [6] [44] such as, performance, security, accuracy etc.
- *A resource* represents a physical or an informational entity.

Goals, tasks, soft-goals and recourses are represented graphically as rounded rectangles, hexagons, clouds and rectangles respectively [6] (see Figure 10).

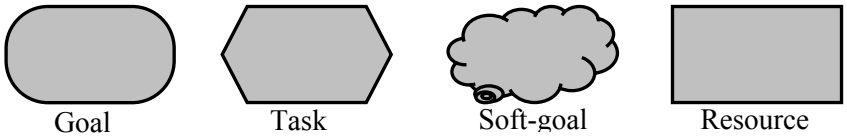


Figure 10: The intentional elements in I* framework

Depending on circumstances, I* framework can either be composed of a global goal model, or a series of goal models distributed amongst several actors [6]. If a goal model includes more than one actor, then the intentional dependency relationships between actors can be represented. Intentional links include means-ends, decomposition, contribution, correlation, dependency and Scenario path links. Graphically, the intentional dependency relationships are represented as shown in Figure 11. The meaning of these intentional links is as follows [6]:

- *Means-ends links*: They describe how goals can be achieved. Each task is connected to a goal by a means-ends link represents an alternative way to achieve the goal.
- *Decomposition links*: They are used to define the sub-components of a task.
- *A Contribution links*: They describe the impact that one element has on another. A contribution can be negative or positive where its extent can be partial (Help or Hurt) or sufficient (Make or Break) according to the concept of satisfying.
- *Correlation links "describe the side effects of the existence of one element to the others"* [6].
- *Dependency links "describe the inter-actor dependency relationships. There can be contributions to a link too"* [6].
- *Scenario path links* describe the sequential order for executing tasks in a scenario.

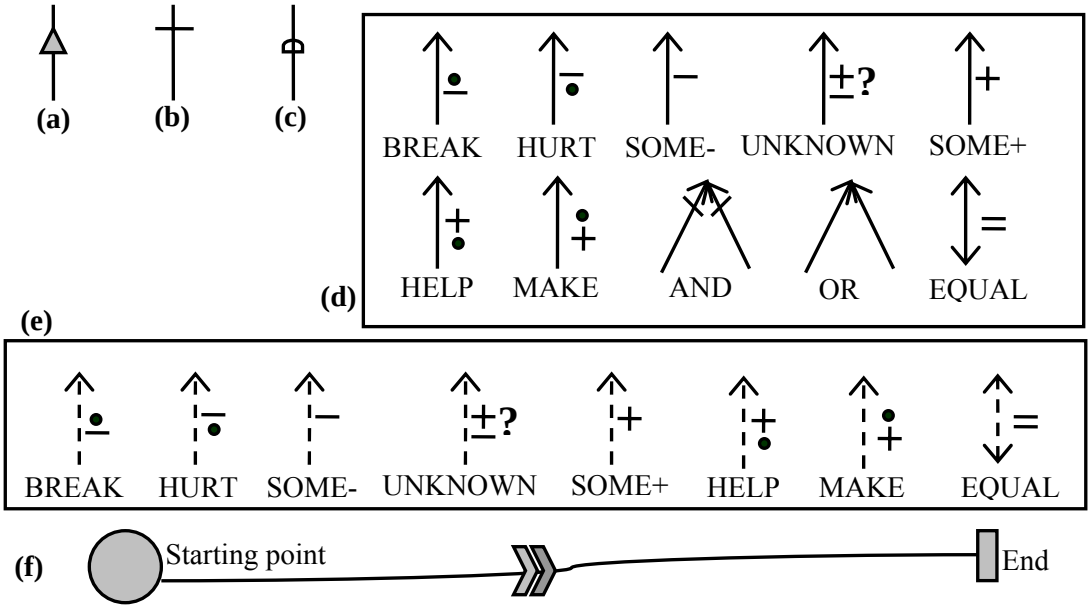


Figure 11: (a) Means-Ends; (b) Decomposition; (c) Dependency; (d) Contribution; (e) Correlation and (f) Scenario path [6].

Finally, I* framework is considered to be interesting since it focuses on high-level security requirements without forcing the modeler to immediately get down to security mechanisms (i.e. Modeling security requirements without requiring an immediate discussion of sizes of cryptographic keys) [6] [47]. Thus, the modeling process helps in clarifying for instance why encryption or authentication is necessary for the system under investigation.

5.3.2.3 MODELING STRATEGIC ACTOR RELATIONSHIPS FOR KERBEROS USING I* FRAMEWORK

The Kerberos environment is used as an example for illustrating the Strategic Modeling technique. Due to the complexity of Kerberos [40], we used a simple Kerberos environment example consisting single Kerberos Server, single Client and single Application Server as shown in Figure 12. For each cooperation, there is an actor depends on another actor to achieve something. For example, the interaction between Kerberos Server and Client from Figure 12 can be interpreted as follows:

- The Client depends on the Kerberos Server to provide him with some services which are the Authentication and ticket service, thus these services are represented as Be Provided [Authentication, Service Ticket]. Since these services represent the Client's interests they are represented using a rounded rectangle.
- The Client depends on the availability of Kerberos Server (i.e. to be available when it is needed). Since the availability is a non-functional requirement it is represented using a cloud with the label Availability [Kerberos Server].
- Kerberos Server depends on the Client to provide some information like his/her user ID and request for the needed service (authentication and/or service ticket) and current time. Thus, this information is represented using a rectangle with the label Be Provided [User ID, Request, Current Time].

In order to provide an idea about how Kerberos Server, Client and Application Server communicate with each other we describe it briefly as follows: cooperation takes place when a Client wants to have specific service from an Application Server. However, the required service (from Application Server) will be provided to that Client only if he/she is authenticated correctly by a Kerberos Server and gets a ticket service from that Kerberos Server. Thus, Client should ask Kerberos Server to provide him with a ticket service in order to get the service from Application Server. Kerberos Server will authenticate the client before providing him with the ticket service.

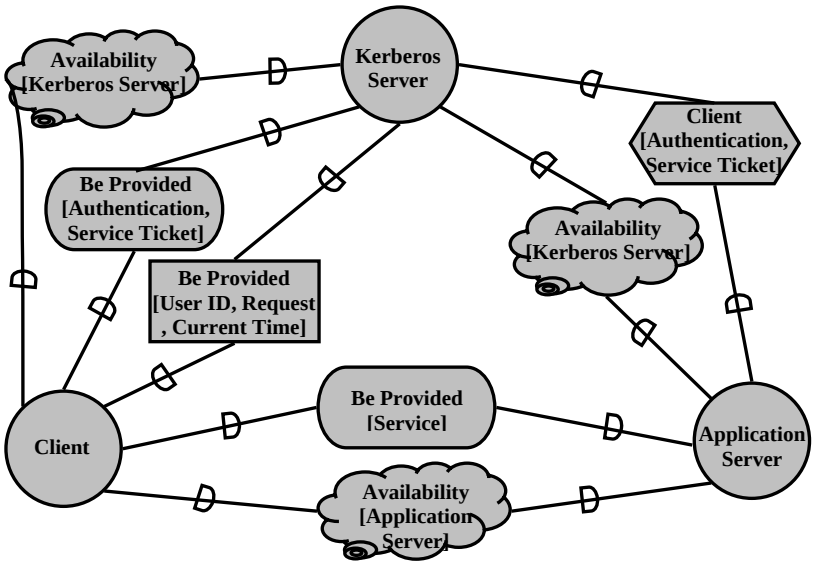


Figure 12: Simple Kerberos environment cooperation model

5.3.3 STRATEGIC MODELING PROCESS

In general, the process of the Strategic Modeling technique can be performed by applying the agent-oriented analysis, goal-oriented analysis and scenario-based analysis for the system under investigation sequentially as follows.

5.3.3.1 AGENT-ORIENTED ANALYSIS

The agent-oriented analysis is based on the idea that abstract role patterns can be modeled and reused. For a particular system, actual system components and functions can be viewed as specializations of generic patterns or the combination of several patterns [6]. Agent-oriented analysis consists of three steps which are:

Step 1: Abstract role dependency analysis - the generic pattern

The first step of agent-oriented analysis is to identify the generic role dependency patterns in the domain [6]. In general, for each dependency link, two main roles can be defined which are User [Service] and Provider [Service] as shown in Figure 13 where *"the cooperation between these two kinds of roles covers the most common relationships between collaborating actors"* [6]. Basically, User depends on the Provider for some User (functional requirements), while nonfunctional requirements arise from both sides as expectations on the other side (e.g. from Figure 13, User [Service] expects from Provider [Service] to protect his/her sensitive information such as user ID, list of requested services etc).

Consequently, in order to identify the dependency relationships between a pair of actors, first requirements engineers have to determine the needed service and add it as a goal, task or resource dependency according to the need of the dependee. Then, security requirements and other quality concerns of the system such as trust, availability, integrity etc. can be added as soft-goals dependencies. In order to add security requirements to the model, Liu et al. [6] suggest the use of catalogue as a supporting reference to elaborate the model according to whether the two actors involved have trust, security or privacy expectations on each other. However, the suggested catalogue was built for P2P applications with emphasize on trustworthiness, credibility, honesty etc. which are not applicable for Kerberos environment; thus we tailored and updated the catalogue to cover the suitable security issues for Kerberos environment. Finally and after establishing the dependency relationships, the roles that attackers could potentially play should be considered. Thus, potential threats/attacks to the system can be modeled as negative contributions of different strengths (BREAK, HURT,

SOME) to dependency links [6]. Potential attackers of the system can be represented as agents with square shadows.

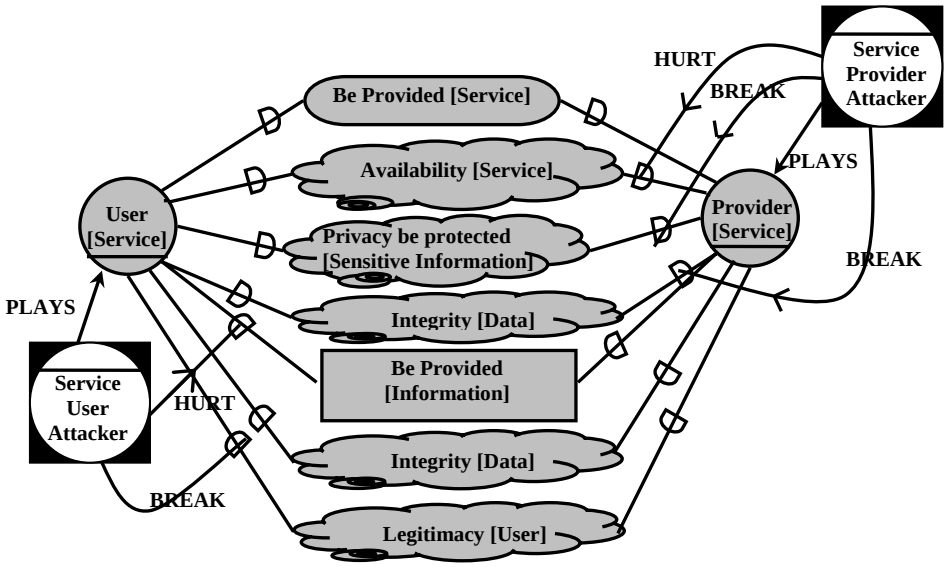


Figure 13: User - Provider relationships

The model in Figure 13 demonstrates the role dependency relationships between User [Service] and Provider [Service] in generic pattern within Kerberos environment regardless who plays the role (i.e. which agent plays the User or Provider role). We can note from Figure 13 the following issues:

- User depends on a Provider to provide some service (BE Provided [Service]).
- User is concerned with the availability of the requested service which should be guaranteed by Provider (Availability [Service]).
- User desires the Provider to guarantee the integrity of the sent data and vice versa (Integrity [Data]).
- User desires the Provider to protect the privacy of his/her personal information such as his/her ID etc (Privacy be protected [Sensitive Information]).
- Provider depends on the User for providing some information (Be Provided [Information]).
- Provider depends on the User for his/her authentication (Legitimacy [User]).

Moreover, Figure 13 shows two potential attackers added to the system with their negative contributions. Service Provider Attacker and Service User Attacker are used to represent attackers' acts to harm the Provider and User respectively. For example, Service Provider Attacker may do harm to the system availability by:

- Flooding requests to the Provider with a consequence of DOS attack to the system [6].
- He/she may try to break the integrity of the transferred data by making some modification.
- He/she may pretend to be the Provider.

Step 2: Role-agent hierarchy in the Kerberos environment

The next step in agent-oriented analysis is to find out the abstract role(s) that each system player is playing [6]. Figure 14 shows a role-agent hierarchy of Kerberos environment. The fist level of the diagram represents the generic roles which apply to all agents within Kerberos environment. Each role within the second level represents specific functionality carried out by roles from third level. Forth level in the diagram specifies the exact agents in the Kerberos environment. In the Kerberos, each agent plays only one role; however in other system, an agent may play (PLAYS link) one or many abstract roles, thereby inheriting the

capabilities and social relationships of the roles [6]. ISA means subtype relationship.

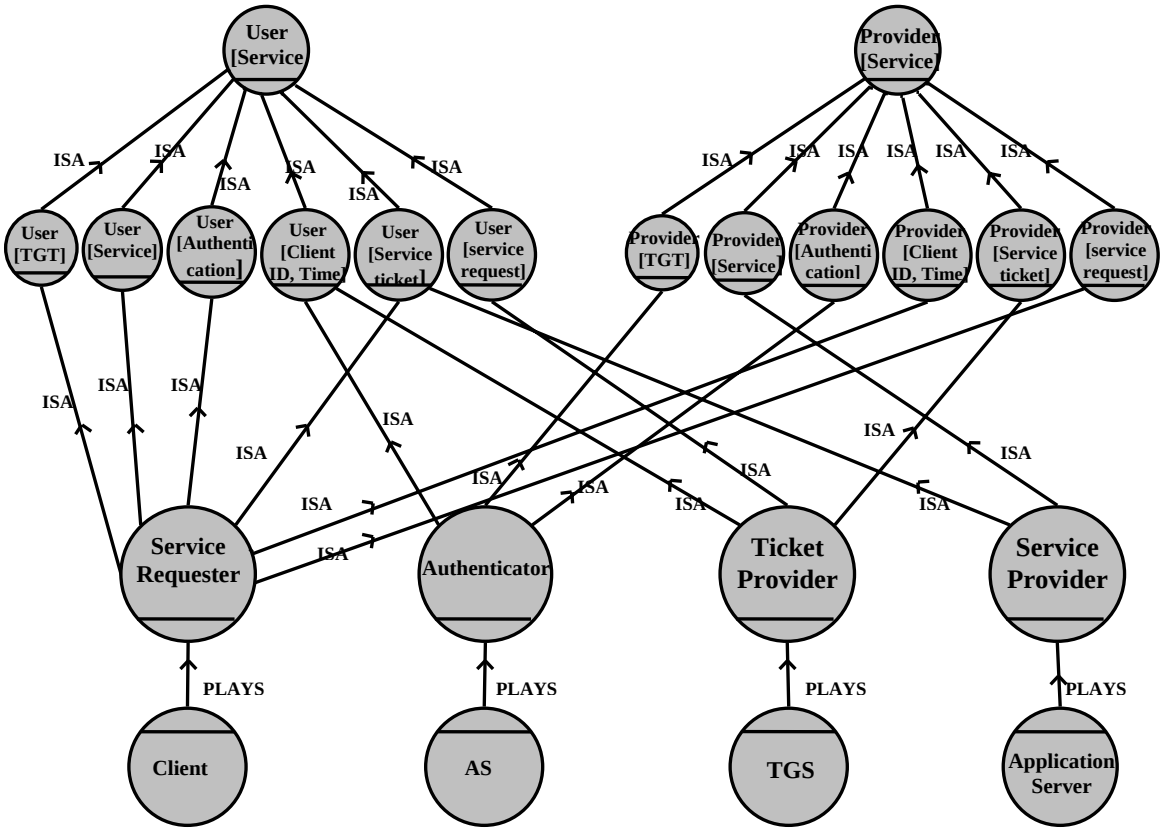


Figure 14: Role - Agent hierarchy for Kerberos environment

Step 3: Dependency derivation along actor associations

The last step in agent oriented analysis is to find out the different pairs for dependency relationships between different roles and agents within the system. Thus, by combining the dependency relationships in Figure 13 and the role hierarchy in Figure 14, we can study pair-wise dependencies [6] between Service Requester, Authenticator, Ticket Provider and Service Provider; these pair-wise dependencies are:

1. Service Requester vs. Authenticator.
2. Service Requester vs. Ticket Provider.
3. Service Requester vs. Service Provider.
4. Authenticator vs. Ticket Provider.
5. Authenticator vs. Service Provider.
6. Ticket Provider vs. Service Provider.

Since each of the six pairs can be considered as User vs. Provider dependencies for a specific service; they can inherit the dependencies between general User and Provider [6] shown in Figure 13. However, the inheritance of dependencies is selective because some of the parameters are substituted with more concrete objects as the analysis moves to lower levels, thus some dependencies may no longer apply [6].

Figure 15 shows the relationships between the main pairs of roles in the Kerberos environment. However, this figure did not cover all the previously mentioned pair-wise dependencies; precisely it covers only the first three pair-wise dependencies because covering all of them will procedure a diagram that can be hard to understand and follow; thus we simplify it as much as it possible. Moreover, Figure 15 can be consider as the final representation for the Kerberos environment and it is consist of combining (1) the first three

pair-wise dependencies from previously mentioned ones; (2) Role - Agent hierarchy for Kerberos environment from Figure 14; and (3) generic pattern for role dependency relationships between User [Service] and Provider [Service] which is represented in Figure 13. Thereby, we can note the following issues from Figure 15:

- How Service Requester vs. Authenticator, Service Requester vs. Ticket Provider and Service Requester vs. Service Provider inherit the dependency relationship from the generic roles shown in Figure 13.
- The red dotted arrows indicate the inherited dependency relationships. For example, the red dotted arrow which connects between blue oval (oval not dotted) and the blue dotted oval which belong to Service Requester vs. Authenticator, shows that Service Requester vs. Authenticator inherit all the dependency relationships from User [Service] vs. Provider [Service] except Legitimacy [User] and Integrity [Data]. Legitimacy [User] not inherited because the main responsibility for the Authenticator is to authenticate user and decide whether he/she is a legitimate user. Integrity [Data] not inherited because the Service Requester will send the data that is requested by Authenticator as a plaintext.
- Both Service Requester vs. Ticket Provider and Service Requester vs. Service Provider inherit the dependency relationship from the generic roles (User [Service] vs. Provider [Service]).
- Client as an agent plays a role as Service Requester and User [Service], AS plays a role as Authenticator, TGS plays a role as Ticket Provider, and Application Server plays a role as Service Provider etc.

A catalogue of security related non-functional requirements in Kerberos environment

Figure 16 shows our vision of the security-related requirements for Kerberos environment. The links between different components (soft-goals, tasks and resources) represent the contributions or side-effects of one component to another [6]. We believe that security for any software system including Kerberos is mainly concerned with CIA concepts. Thus, CIA elements are presented in high level and linked together with *"And"* relationship which indicates that Availability, Confidentiality and Integrity are the main components that build secure software. Then, the most related popular defense mechanisms that enhance each of the CIA components are linked to the specific component using *"Help"* relationship. The dotted line indicates that detect and defense mechanisms has a positive side-effects regarding system availability, but it is not sufficient to fulfill system availability. Other links describe the impact that one element has on another (all of the links indicates that the impact is positive and partial).

However, this catalogue did not cover all the related issues for security; instead it was built and used to clarify some issues regarding the Strategic Modeling technique. In reality, the catalogue can be elaborated to cover another security-related requirements such as trust, privacy etc. Also, other nonfunctional requirements might be added such as performance, reliability etc.

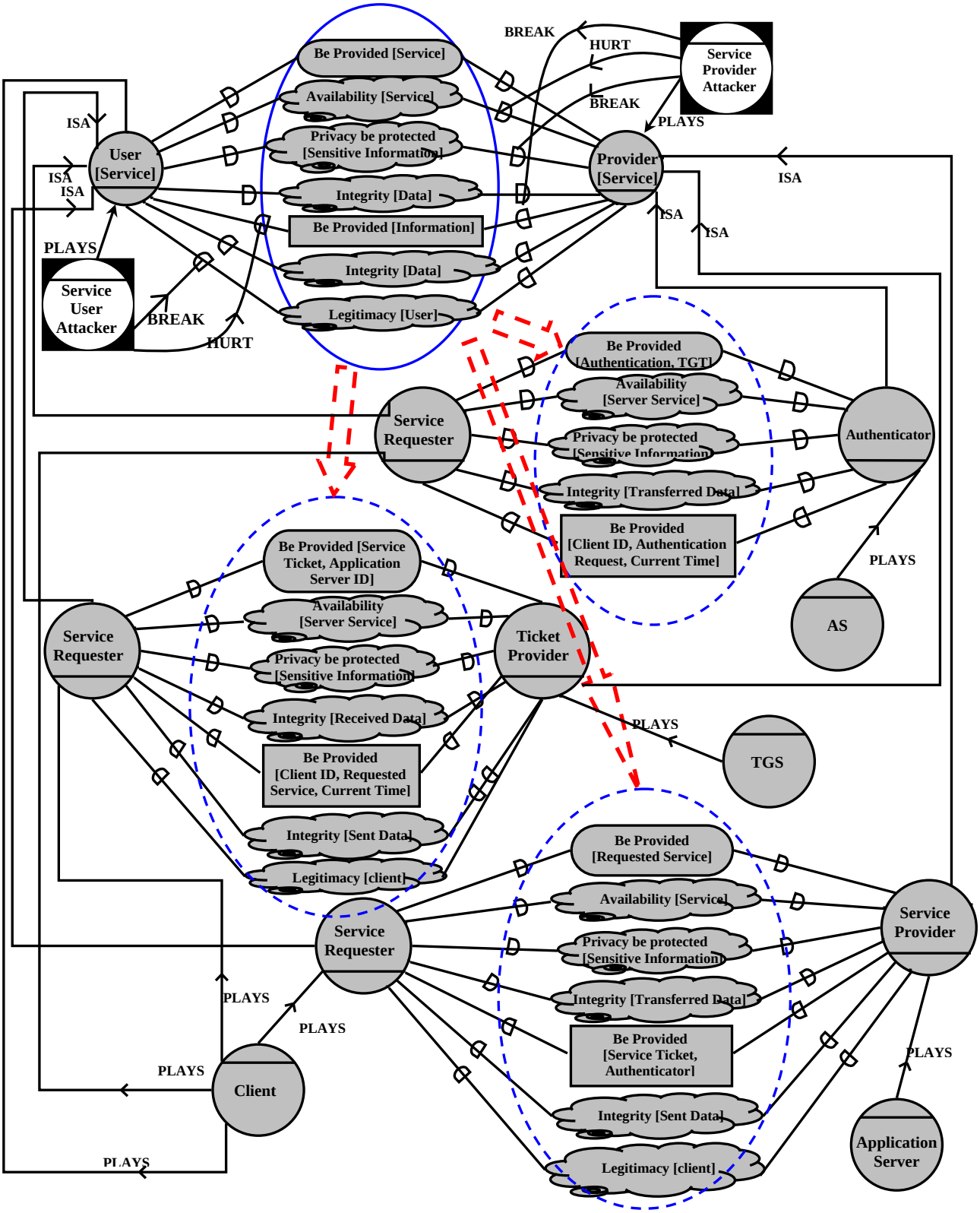


Figure 15: Dependency derivation

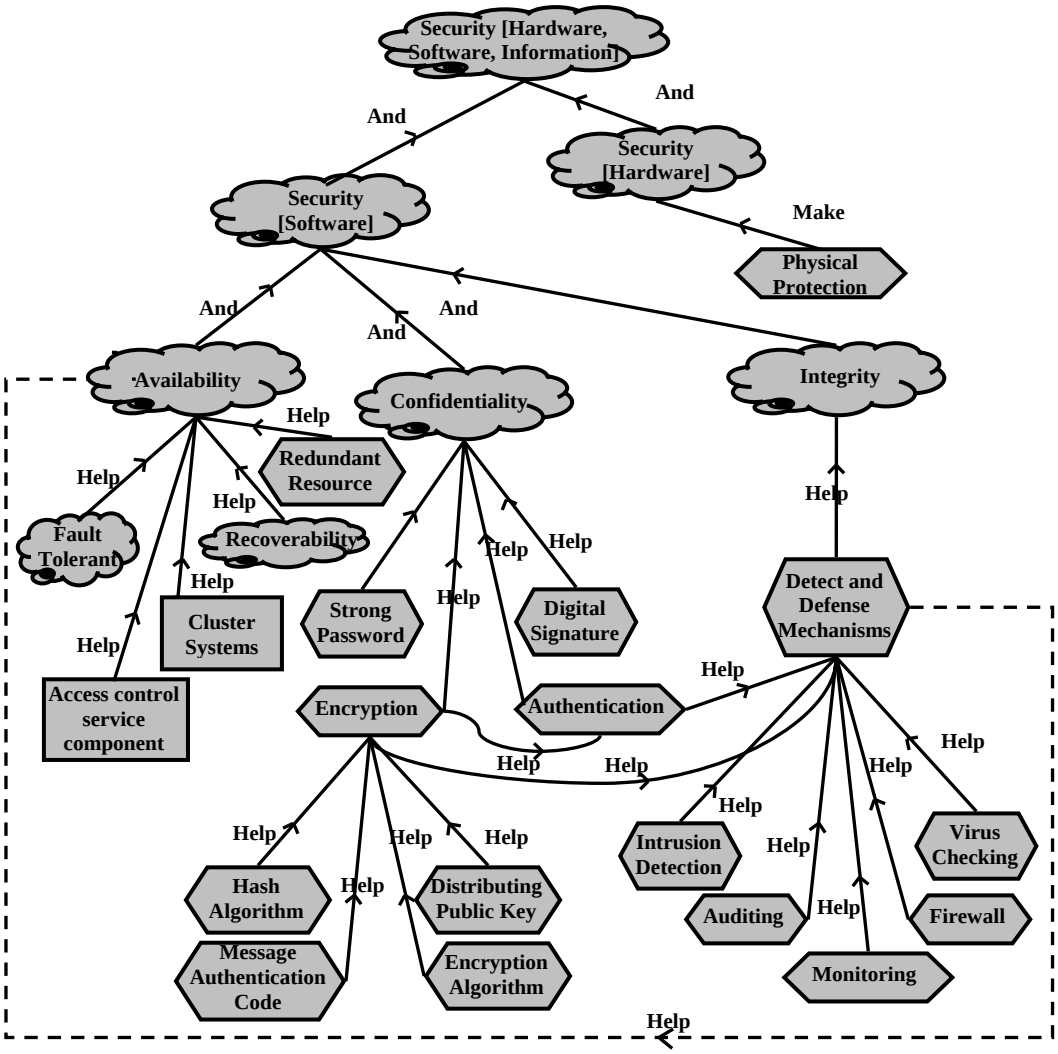


Figure 16: Catalogue for security related requirements for Kerberos environment

5.3.3.2 GOAL-ORIENTED ANALYSIS

Goal-oriented analysis aims to analyze the goal dependency and elaborate goal refinement to show how such refinements within an actor can lead to new dependencies among actors [6]. In order to explain goal-oriented analysis, the goal dependency (Be Provided [Requested Service]) between Client (Service Requester) and Application Server (Service Provider) is elaborated in more detail. Thus, before providing requested service to the Client, Application Server needs to assess both Legitimacy of the User and Integrity of the service ticket. Thus, Legitimacy and Integrity become sub-softgoals for the task (Provide [Requested Service]). In other words, Client should prove his/her Legitimacy to the Application Server as well as the Integrity of the service ticket. In Kerberos environment, Application Server depends on AS to assess Client legitimacy (i.e. Client already authenticated by AS) and depends on TGS to assess the Integrity of the service ticket. Thus and based on both the common sense knowledge (the security catalogue) and the current situation, Application Server considers whether the sub-softgoals (Legitimacy and Integrity) can be satisfied. Figure 17 models the structure of the goal (Be Provided [Requested Service]) from Application Server point of view.

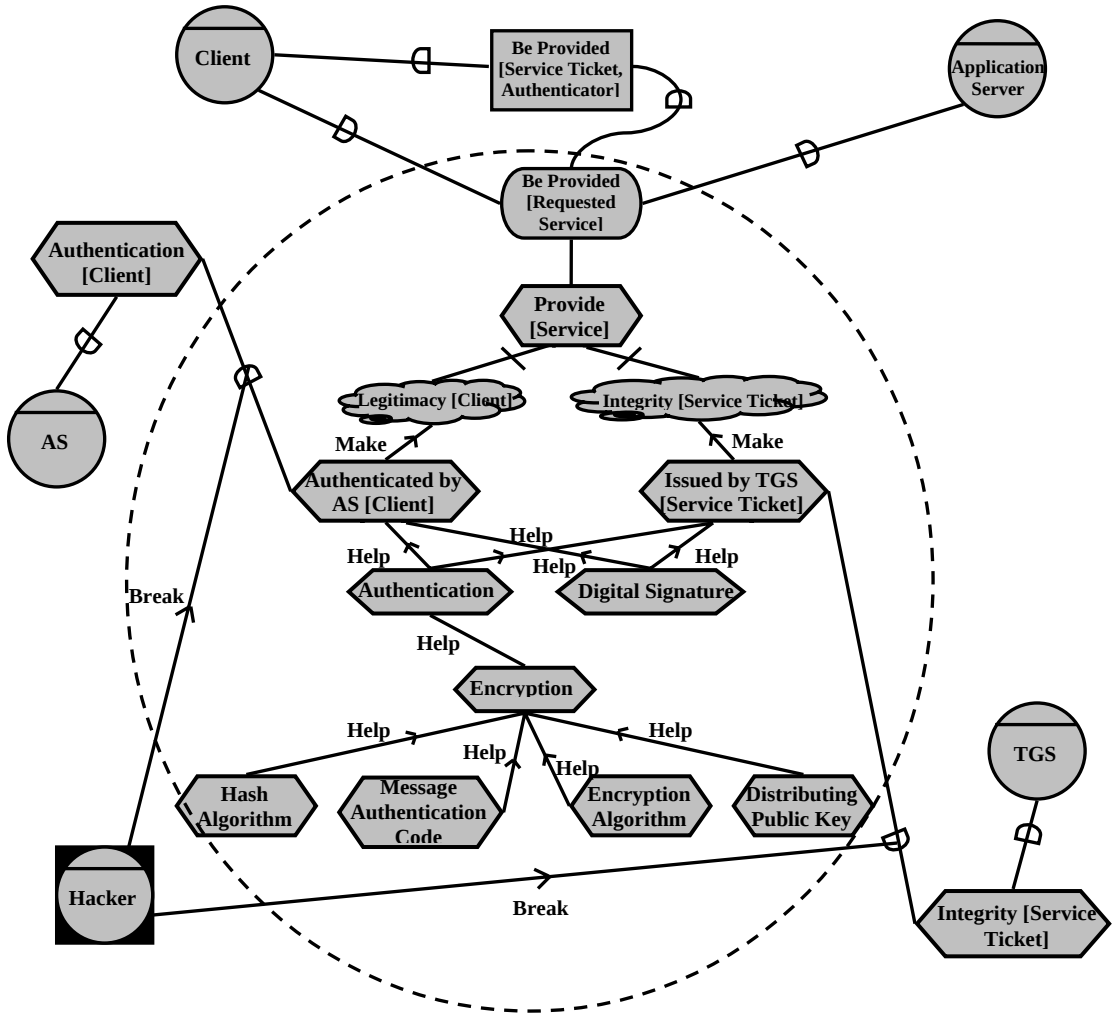


Figure 17: Legitimacy and integrity analysis for the goal (be provided [requested service]).

5.3.3.3 SCENARIO-BASED ANALYSIS

During scenario-based analysis the focus will be on the elaboration of both agent-oriented analysis and goal oriented analysis. Thus, analyst may focus on the intentions of attackers and the potential consequences of their attack. Dependencies between different agents in Kerberos environment are highly related to each other (e.g. Client depends on AS for providing authentication service and providing ticket granting ticket (TGT) in order to communicate with TGS; also TGS depends on AS to confirm that the client is authenticated). Thus, different kind of attacks might be analyzed such as man in the middle attack, replay attack, DOS attack etc.

5.3.4 ADVANTAGES VS. DISADVANTAGES

One of the main advantages of the Strategic Modeling Technique is the use of I* framework for modeling which offers [6]:

- Structural representation for the intentional relationships between two actors or more a as well as within two actors or more.
- Structural concepts such as intentional agents, roles, goals/soft-goals, etc.
- I* framework encourages and facilitates the analysis of security-related issues within the full operational and social context of relevant actors (e.g. Internet provider might be added as relevant actor in some circumstances).

Thus, the Strategic Modeling technique seems to be suitable techniques for modeling and analyzing security requirements for the following reasons:

- The Strategic Modeling technique helps to model security and other nonfunctional requirements for any system that runs in the Internet environment since it covers *"the operational procedures, potential attacks, countermeasures against perceived threats as well as factors not directly related"* [6].
- Security requirements can be taken in consideration from the beginning of the development life cycle.
- The Strategic Modeling technique covers both modeling and analyzing security requirements with more emphasis on the analysis part. Thus, this technique focuses on the high-level of security requirements without forcing the modeler to immediately get down to security mechanisms instead it shows why encryption or authentication, for example, is important for the specific case as shown in Figure 17.
- Security requirements can be modeled without adding new notations to the I* framework. However, produced document that includes diagrams might be understood easily by people who are familiar with the notations of the I* framework; but it mostly will be considered as hard to be understood by people who are not familiar with the notations of the I* framework especially if the investigated system includes many agents and roles with much dependences between them.
- The Strategic Modeling technique covers the main security concepts (CIA) since a catalogue for security related requirements of the system under consideration would be produced and used as a reference for the analysis process.

However, the process performance might be considered as a critical issue when using the Strategic Modeling technique since building diagrams and performing analysis can be considered as a time consuming process especially if the system includes many actors that depend on each other and the analysis is conducted in more elaborated way. Thus, the process performance depends on (1) number of actors within the system, (2) their dependency on each other and (3) depth of the analysis and its elaboration way. Therefore, a tradeoff between these things should be taken into consideration to manage the process performance. Anyhow, using a tool such as OME [57] might help in accelerating the process performance. Nevertheless, we think that the possibility of using the Strategic Modeling technique in practice is high because it generates a structural representation for the security requirements within system under investigation.

Another disadvantage of the Strategic Modeling technique is not specifying the attacker capability. Thus, a description of the attacker including his/her resources, skills and objectives can be added during the analysis process.

5.3.5 STAKEHOLDER'S INVOLVEMENT

There are different types of stakeholders that might be involved in order to perform the Strategic Modeling technique. Those stakeholders are requirements engineers, users, security experts and developers whereas their duties are the same as mentioned in section 5.1.2.3.

5.4 ATTACK TREES TECHNIQUE

Attack Trees technique provides a formal, methodical way of describing the security of systems, based on varying attacks [49]. Essentially, different ways of attacking a system are represented in a tree structure where the attack goal is represented as the root node and different ways of achieving that goal are represented as leaf nodes.

Basically, this technique helps to identify and understand different ways that might be used to attack the system; therefore assisting the requirements engineer to define system security requirements as well as helping the designer and the architect to choose the suitable countermeasure(s) to thwart each attack pattern.

5.4.1 ATTACK TREE STRUCTURE AND NOTATIONS MEANING

Attack trees consist of two types of nodes: *AND-nodes* and *OR-nodes* [49] [50].

- An AND-node represents different steps or events to achieve the same goal (i.e. all children of an AND-node should be executed to reach the goal represented by the AND-node).
- An OR-node represents the alternative ways to achieve the goal (i.e. execution of any child of an OR-node suffices to reach the goal of the OR-node).

Attack trees can be represented graphically or textually as shown in Figure 18. Graphically, an AND-node is represented by the curved line that connects the lines drawn from the root-node to its children while any non-leaf node without this curved line is an OR-node. Textually, any (OR) that adjacent to the end of a sub-goal or a leaf indicates that this item is connected to the next item from the same level by an OR relationship (e.g. the OR adjacent to the end of leaf1 indicates that the relationship between leaf1 and leaf 2 is an OR relationship); adjacent AND is addressed the same way.

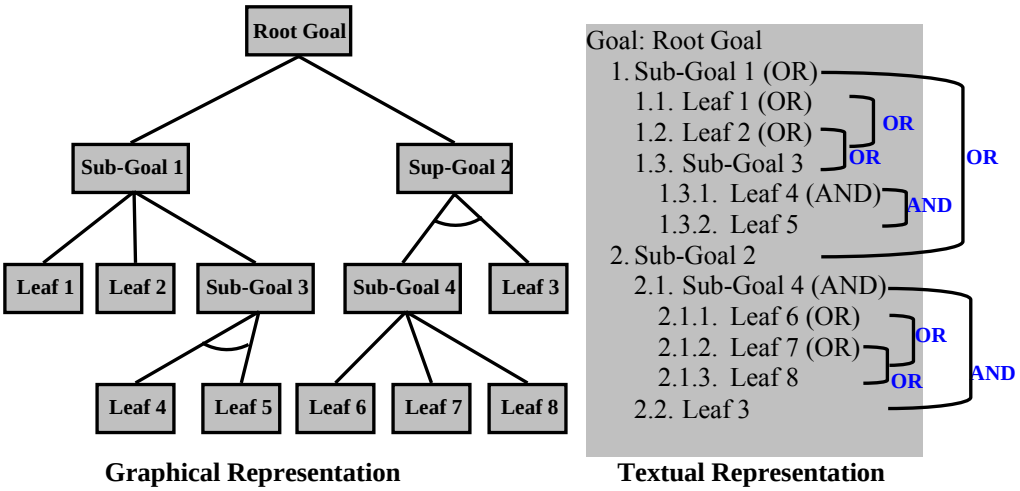


Figure 18: Attack tree representation

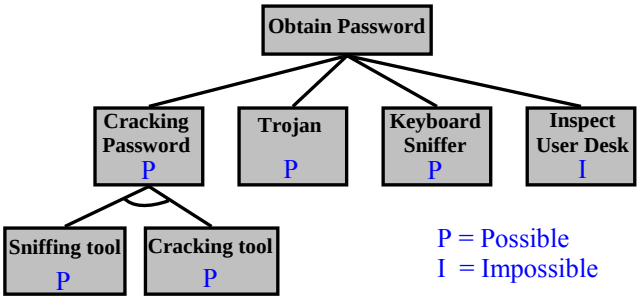
Each leaf node represents specific attack action, event or step that must be achieved in order to accomplish the goal [49]. Thwarting any attack can be achieved by making the attack goal impossible to be accomplished and this can be controlled based on the child nodes type (AND/OR-Nodes). In other words, if the attack goal is an AND-node, then the attack can be prevented by countering at least one sub-goal. On the other hand if attack goal is an OR-node, then all sub-goals must be countered. Countering can be performed by implementing one or more security countermeasures.

5.4.2 CREATING ATTACK TREES

In order to create attack trees, several steps should be followed in a repetitive way. These steps can be summarized as follows [49]:

1. Identify the possible attack goals.
2. Choose a goal from step 1 and set it as a root node.
3. Identify different kinds of attacks against the chosen goal from step 2 and add them to the tree. This can be achieved by first decomposing the parent node into children nodes (sub-attacks) and then determining the relationship between the children nodes (AND/OR).
4. Repeat step 3 downwards the tree until there is no attacks can be identified anymore.
5. Refine attack tree. Let someone else review the tree and consider his/her suggestions regarding the attack tree (e.g. add any new nodes).

Figure 19 represents a simple attack tree in which the goal is to obtain the user password. An attacker may try to crack the password, install a Trojan program on the user's computer, trick the user to install a keyboard sniffer program or he/she may physically inspect the user desk hopefully to find the password written on a piece of paper. Those are some actions that might be done by the attacker to achieve his/her goal; thereby all of them are represented as leaf nodes except cracking password which is represented as AND-node that requires both sniffing tool and cracking tool to accomplish the cracking process and getting the password in a plaintext. The meaning of the associated value Possible/Impossible is described in more detail in the next section since they belong to the analysis phase. Regarding Figure 19, it is worth to mention that this attack tree which addresses specific security requirement *"system should protect the user password"* is just a simple attack tree, and an incomplete one. On the other hand, Figure 20 presents a more formal attack tree of a part of the popular PGP e-mail security program. Since PGP is a complex program the tree is also complex, thereby it is easier to present it textually than graphically [49].



Goal: Obtain Password
1. Cracking Password (OR)
   1.1. Sniffing tool (AND)
   1.2. Cracking tool
2. Trojan (OR)
3. Keyboard Sniffer (OR)
4. Inspect user desk

Figure 19: Simple attack tree for obtain password

In order to generate and build a complete attack trees, different participants should be involved such as requirements engineers, analysts, architects, security experts etc. Since building a complete attack trees takes time, the task for building an attack tree can be assigned to one of the involved participants while each of the other participants can try to build another attack tree. Then, each of them may review and update the attack trees built by others (e.g. add new kind of attacks). Thus, using this way of work may help in saving the project time. However, there is no guarantee of completeness [49] [50]. In other words, even if the attack trees are built in incremental manner and refined by different participants, there is still a chance to forget something.

Generally, attack trees can be used to model and analyze security requirements. However, building and analyzing attack trees can be done in two stages. The first stage is the modeling stage and it is mainly conducted by the requirements engineer in which a detailed description of different attacks is not required (i.e. requirements engineer needs to define different sort of threats against the system). The second stage is the analysis stage and it is mainly conducted by the analyst who may need to investigate different kind of attacks as well as associated attributes and their values in more detail in order to specify and analyze possible attacks and figure out the countermeasures that might be used to thwart them.

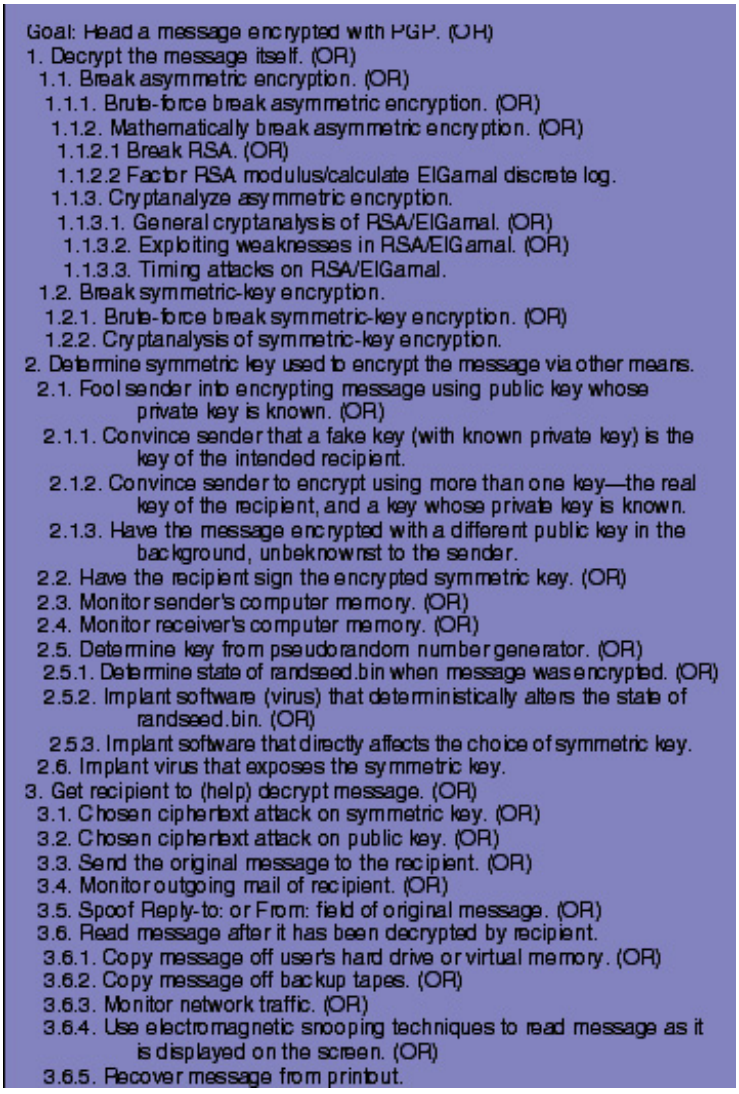


Figure 20: Attack tree against PGP system [49]

5.4.3 ANALYZING ATTACK TREES

Once an attack tree is modeled, the analyst can assign some attack attributes with their values for all leaf nodes [49] [50]. The attributes values can be Boolean, numerical or a combination of them. The Boolean values can be such as possible/impossible, easy/difficult, legal/illegal, special equipment required/no special equipment required etc. while the numerical values can be such as the cost of performing the attack, the needed time etc.

After assigning the attack attributes and their values to the leaf nodes, analyst should propagate the attributes values upward the tree to the root node (i.e. the value of a non-leaf node is synthesized from the values of its children) [49] [50]. This should be done according to the logical aggregation (AND/OR). Generally, in order to calculate the attack cost for an attack tree, the analyst should calculate the attack cost for all the non-leaf nodes whereas OR-nodes have the attack cost of their cheapest child; while AND-nodes have the attack cost as a summation of the attack cost for their children. In case of Boolean attributes, take possible/impossible for instance, the value of an OR-node is possible if any of its children is possible and it is impossible if all of its children are impossible. On the other hand, the value of an AND node is possible if all of its children are possible while it is impossible if one or more of its children are impossible.

In Figure 19, we use the Boolean value possible/impossible to indicate the possibility of performing the attack task. Thus, the leaf node *"inspect the user desk"* is considered as

impossible since the user is the only one who have the key for his/her office; thus it is impossible for an attacker to get inside the user office and look for the written password. However, inspecting the user desk is possible if the attacker is willing to break the office door. On the other hand, the other alternatives might be possible to occur; thus there is a need to find suitable countermeasures to overcome them. Anyhow, deciding values for the attack attributes is not an easy task as it may seem and depend on many factors. Therefore, the analyst should decide the attributes values carefully, since these values are used to calculate the value of the root node which might be used to specify the occurrence probability of this attack. One suggestion to support the attributes values is by linking the attacker attributes to the attack trees to provide some sense for the calculated values. Also, specifying the precondition(s) to achieve certain task might help in determining the probability of that attack. For example, in Figure 19 in order to perform the keyboard sniffing attack, an attacker needs to trick the user to install a keyboard sniffer program, thus he/she can get the user password.

Once the attack tree is analyzed, it can be used to make requirements and security decisions for the system under investigation [49]. Take the cost attribute for instance, if the attack costs more than the expected benefit, then this attack might be classified as *"most likely will not occur"*. However, if the attack is easy to be performed and may result in benefit, then this attack should be investigated in more detail to determine the suitable countermeasure(s). On the other hand, taking a decision to classify an attack as either likely or not likely to occur is different for different circumstances and depends on the assumed knowledge about the attackers. Attackers have different level of skills and financial support. Thus, for instance if the system is worried about an organized attack, the analysts have to worry about expensive attacks [49]. Moreover, using different attributes with the attack tree helps to determine the cheapest attack that does not require special equipment, the attack that does not require high skills etc [51]. Using some tools such as SecurITree can help in generating wide range of queries depending on the used attributes [51].

In addition, during the analysis of the attack trees, analyst may find out that the key length of the used encryption algorithm does not really matter if an attacker successes in tricking a user to install a keyboard sniffer program or if he/she can modify the system on the victim's hard drive. Attackers find that these kinds of attacks are much easier than breaking the key of an encryption algorithm such as RSA [49].

On the other hand, it is worth to mention that the value of the attack trees, attackers characteristics (skills and financial support), or even attack trees construction might be changed during the analysis process for one reason or another (e.g. finding that one way of attack is missing, changing the value of some assigned attribute). In these cases, the new change should be reviewed and applied; also the missing attack technique or process should be added to the attack tree. This requires recalculating the value of the root node.

5.4.4 ADVANTAGES VS. DISADVANTAGES

The Attack Trees technique seems to be a suitable technique for modeling and analyzing security requirements with emphasize on the analysis part. Consequently, there are many advantages of this technique as well as some disadvantages or, specifically, weak-points.

One of the main advantages that make Attack Trees technique valuable is the reusable form of the produced attack trees [49] which helps in saving time. Thus, once a tree for a particular goal is completed it can be reused whenever that goal is used in other attack trees (i.e. the attack tree for a specific protocol such as TCP can be reused for any system that uses TCP). However, in some cases the attack trees might be modified to be suitable to the new system. Reusable form may motivate the analyst to create attack patterns as a generic representation of malicious attack which commonly occurs in specific contexts; also he/she may create general patterns that describe the attackers' characteristics. Thus, referring to these patterns when there is a need and then making the necessary updates.

Another advantage of the Attack Trees technique is that it can be used with other techniques for modeling and analyzing security requirements such as the Misuse Cases, Abuse Cases, Data Sensitivity and Threat Analysis, and Strategic Modeling techniques. For instance, combining Attack Trees technique with the Strategic Modeling technique can lead to improve the analysis of the security requirements. For example, referring to Figure 13, the analyst may create an attack tree with the root goal *"break data integrity"* which shows different attack ways to achieve that goal. Also, the Misuse Cases technique and/or the Abuse Cases technique can be merged with the Attack Trees technique in order to provide more information regarding specific attacks such as preconditions, post-conditions and attacker characteristics. In addition, the Data Sensitivity and Threat Analyses technique can be used to determine whether there is a need to create an attack tree to protect certain piece of information within the system.

Defining and analyzing possible attacks against the system in a structured way is another advantage of using the Attack Trees technique. This structure is expressed in a node hierarchy, allowing the analyst to decompose an abstract attack or attack goal into a number of more concrete attacks or sub-goals [50]. Also, this structure can help in addressing different sorts of attacks against CIA concepts, thus covering the main aspects of the security. In addition, the existence of some tools such as SecurITree [51] from Avenaza for building attack trees might be considered as another advantage for using the Attack Trees technique since this tool and the like will help in building, updating, and managing the built attack trees easily.

On the other hand, two main weak-points regarding the Attack Trees technique exist. First: assessing quantity and quality of the attached detail about the attackers and the attack attributes and their values. Second: there is no guarantee of completeness. We think that both risks are expected since the Attack Trees technique requires a certain mindset [49] and depends on the number of participants involved in the process, the analyst experience, his/her knowledge regarding different kinds of attacks and his/her guesses regarding attackers. Both risks can be minimized by involving different participants as many as possible. Also, creating an attack tree and attackers patterns can help in minimizing these risks. Furthermore, building attack trees with more detail and analysis for large systems might be considered as time and budget consuming process. However, this depends on many factors such as the nature of the project, the available time and budget, the number of involved participants etc. Nevertheless, the simultaneously work to create and refine attack trees can help in speeding up the process.

5.4.5 STAKEHOLDER INVOLVEMENT

There are different types of stakeholders and participants that might be involved in order to create and analyze the attack trees. Those stakeholders are requirements engineers, users, architects, security experts and developers. Their duties are the same as mentioned in section 5.1.2.3 and section 5.2.3.

Chapter 6

DISCUSSION AND MOTIVATION

In this chapter, we discuss in general the similarities and the differences between the different techniques that are used to model and analyze security requirements, namely the ones that are mentioned in Chapter 5. In addition, we present our point of view regarding how to fit security requirements into the conventional requirements engineering using the discussed techniques in Chapter 5. Moreover, we offer some general suggestions and recommendations regarding the modeling and analysis techniques of security requirements as well as the development process for a secure system. Finally, we mention the missing aspects according to our perception.

6.1 SIMILARITIES AND DIFFERENCES BETWEEN DIFFERENT TECHNIQUES

To give a general overview of the different techniques that have been discussed in Chapter 5, we conducted three kinds of comparisons between these techniques according to different criteria and based on our point of view.

The first comparison is concerned with the following criteria:

- *Covering the security concepts (CIA):* whether each of these techniques covers the security concepts (CIA); this is done according to the description that has been carried out in Chapter 5.
- *Performance:* the overall performance of the technique (the needed time to use the technique). This is done based on (1) the process of the technique; (2) the existence of tools that might be used to accelerate the process; and (3) by relatively comparing the time that we spent to build the used example in Chapter 5 with some real project from our experience.
- *The use in practice:* we estimated the possibility of using the technique in practice according to whether the technique uses standard notations such as UML, the use of similar technique in practice (e.g. Abuse Cases and Misuse Cases techniques are similar to the Use Cases technique), whether the technique is easy to use and the difficulty to understand the used notations. Moreover, all other criteria (covering the security concepts, performance and trusted assumption) are taken into our consideration when we measure the possibility of using the technique in practice.
- *Trusted assumption:* whether the technique uses trusted assumptions such as an assumption about the attacker characteristics or communication between two components or more of the system.

Thus, we used high, medium-high, medium, medium-low and low scales to rate the previous criteria except the trusted assumption criterion which takes either Yes or No. The result of this comparison is shown in Table 3.

Criteria / Technique	Covering the security concepts (CIA)	Performance	The use in practice	Trusted assumption
Abuse Cases	*Medium*	*Medium-high*	*High*	*Yes*
Misuse Cases	*Medium*	*Medium-high*	*High*	*Yes*
Data Sensitivity and Threat Analyses	*Low*	*Medium*	*Medium*	*Yes*
Strategic Modeling	*Medium-low*	*Medium*	*Medium-low*	*No*
Attack Trees	*Medium-high*	*Medium-high*	*High*	*Yes*

Table 3: Comparison between some of the modeling and analysis techniques of security requirements.

We can note from Table 3 the following issues:

- The Abuse Cases and Misuse Cases techniques are considered to be the same. Both of them cover the security concepts. However, the depth of covering the security concepts depends on (1) the investigation level: whether the abuse/misuse cases are defined in general and whether the analysis of the defined cases has been done in a deep level. (2) Number of defined cases to cover Conditionality, Integrity and Availability. Both the investigation level and the number of defined cases are highly related to each other. Thus, each defined cases should be investigated in a good way. However, the structure of the abuse/misuse cases can not provide a deep level of investigation unless we use some templates to describe the abuse/misuse cases (i.e. describing the preconditions, post-conditions, different paths etc.). Also, the security concepts are modeled in an acceptable level comparing to the other techniques. Thus, by taking all of these things into consideration the value of covering the security concepts is considered to be *medium*. Performance for both of them considered to be *medium-high* since the used idea for creating both of them is similar. Also, there are some similarities in the used process in both techniques and the elapsed time to build the used examples in Chapter 5 was almost equally. Moreover, there is a possibility to use the available design tools such as Rational Rose can help in speeding up the process. The possibility to use any of them in practice is considered to be *high* for many reasons such as both of them were built upon the Use Cases technique which is used widely in practice to elicit, model and analyze functional requirements. Also, both of them use standard notations (UML), easy to use and easy to understand. Both of them use trusted assumptions about attackers, communications etc.; thus using trusted assumption is considered to be *Yes*.
- For the Data Sensitivity and Threat Analyses, the depth of covering the security concepts depends on the developed antipatterns which are generally built in a generic form without putting much effort in investigating the security concepts in detail. Also, modeling security concepts using this technique is done in a superficial way. Thus, the value of covering the security concepts is considered to be *low*. Performance is considered to be *medium* since the main task of this technique is to understand the value of data that should be handled by the system as well as classifying the data in different categories which may require conducting several meeting to come up with an acceptable agreement for different stakeholders of that classification. Also, it is worth to note that conducting such meeting is a time consuming process and it will affect the performance negatively. Since classifying data into different categories is the most important issue within this technique and the output of using this technique is easy to understand, the possibility of using this technique is considered as *medium*. Finally, since some assumptions about attackers are used in this technique, the trusted assumption criterion is considered to be *Yes*.
- For Strategic Modeling, the depth of covering the security concepts is done in a high level (i.e. this technique focuses on the high-level of security requirements without

forcing requirements engineer to immediately get down to the security countermeasures). However, the analysis of the security concepts can be done in depth by involving the use of the security concepts. Thus, by taking a tradeoff between the analysis level and the modeling level, the value of covering the security concepts is considered to be *medium-low*. Performance is considered as *medium* since the modeling and analysis processes depend on a number of actors within the system, their dependency, depth of the analysis and the elaboration of the analysis. Also, the process of the technique consists of several steps which need more time in order to produce valuable figures that cover the main actors in the system and their dependencies. However, using a tool such as OME [57] might help in accelerating the process performance. The possibility to use in practice is considered to be *medium-low* since using this technique requires special training in order to use it in an effective way. Also, it is not easy to understand the produced figures unless the reader is familiar with the I* notations. Finally, trusted assumptions take the value *No* since there is no trusted assumptions were used. However, if the attackers' capabilities are used then the assumptions about attackers will be used and the value of the trusted assumption might be changed to *Yes*.

- For the Attack Tree, covering the security concepts can be done in various levels in both modeling and analysis phases. Also, by taking into consideration that there is no guarantee of completeness and quality of the attached attributes and their values in the analysis phase, the value of covering the security concepts is considered to be *medium-high*. Performance is considered to be *medium-high* since this technique offers the reusability form. Also, using tools such as SecurITree can help in accelerating the process of the technique. Due to its performance, covering security concepts in varies level, easy to use and easy to understand the process outputs, the possibility of using the Attack Trees technique in practice is considered to be *high*. Finally, trusted assumption takes the value *Yes* since some assumptions about the attackers are used during building and analyzing the attack tree.
- The assigned values for covering the security concepts criterion have the following meaning: *low* means that the technique covers these concepts in a superficial way, *medium-low* means that the technique covers these concepts in a level which is less than medium (not so accepted level), *medium* means that the technique covers the security concepts in a medium level (accepted level), *medium-high* means that the technique covers the security concepts in a level which is more than medium (more than the accepted level, but not so much) and *high* means that the technique covers the security concepts in a high (deep) level (very accepted level).
- The assigned values for the performance criterion have the following meaning: *low* means that the performance is time consuming, *medium-low* means that the performance is less than medium (not so accepted performance), *medium* means that the performance located in an accepted level of performance, *medium-high* means that the performance located in a level which is more than medium (high performance but not so much) and *high* means that the performance is high (very accepted level).
- The assigned values for the use in practice criterion have the following meaning: *low* means that the technique is used in low level (very few), *medium-low* means that the use of the technique is less than medium (not so accepted level), *medium* means that the technique is used in an accepted level, *medium-high* means that the technique is used in a level which is more than medium (high use but not so much) and *high* means that the technique is used in a high level (very accepted level).
- Generally, all of these techniques use trusted assumptions except the Strategic Modeling technique. The used assumptions are mostly about the attackers' characteristics as well as the likelihood of the attacks. Also, all of these techniques can cover the security concepts but in a different levels (depths). In addition, performance of these techniques is highly affected by the size of the system under consideration, the number of participants involved in the modeling and analysis processes, whether

supporting tools are used during the process, creativity of the requirements engineers and system analysts etc. Finally, the possibility to use in practice is affected by the other criteria.

- It is worth to mention that using the trusted assumptions which are described, analyzed and assessed in a good way will affect using the technique positively. On the other hand using the trusted assumptions which are described, analyzed and assessed badly will affect using the technique negatively.

The second comparison is concerned with whether each of these techniques covers the modeling and the analysis phase together as well as the used notation for the modeling phase and sub-processes for the analysis phase. This comparison based on the description of these techniques which has been carried out in Chapter 5. Thus, Table 4 holds the result of this comparison. Finally, the third comparison shows the weakest and the strongest point for each of these techniques as shown in Table 5.

Technique	Modeling	Analysis
Abuse Cases	UML notations + templates filled using natural language descriptions.	Descriptions of the attacker's characteristics, assumptions and abuse case diagrams and descriptions.
Misuse Cases	Modified UML notations + templates filled using natural language descriptions.	Descriptions of the attacker's characteristics, assumptions and misuse case diagrams and descriptions.
Data Sensitivity and Threat Analyses	There is no formal approach, just produce some general patterns in a natural language description.	Understanding the sensitivity of the data within the system as well as the impact if the data is disclosed or altered by an unauthorized person.
Strategic Modeling	I* framework notation.	Three kinds of analysis were used which are: Agent-oriented analysis Goal-oriented analysis Scenario-based analysis
Attack Trees	Tree structure represents different ways of attacking the system. However, each attack tree can be represented graphically or textually.	The process consists of two main steps. First, assigning the attack attributes and their values to all leaf nodes. Second, propagate the attributes values upward the tree to the root node.

Table 4: Comparison between the modeling and the analysis phases for some of the modeling and analysis techniques of security requirements.

Technique	The weakest point	The strongest point
Abuse Cases	Deciding the number of abuse cases that should be developed within the system.	Easy to understand and it can be used to increase awareness of the system stakeholders regarding the security features and system vulnerabilities.
Misuse Cases	Deciding the number of misuse cases as well as introducing the additional use cases and their relationships with the other use and misuse cases.	Presenting use and misuse cases together.
Data Sensitivity and Threat	Informal process.	Understanding the real value of the data we need to protect.

Analyses		
Strategic Modeling	Not specifying the attacker capability.	Formal analysis process.
Attack Trees	Assessing quantity and quality of the attached detail about the attackers and the attack attributes and their values.	Reusable form.

Table 5: Comparison between the weakest and the strongest point for some of the modeling and analysis techniques of security requirements.

Finally, Table 6 presents our vision regarding the levels (depths) that are offered by these techniques to cover the modeling and the analysis phases. In other words, to which kinds of systems these techniques are suitable.

We can note from Table 6 that the Abuse Cases and Misuse Cases techniques have many similarities regarding the modeling and the analysis phases. This confirms what is previously mentioned in section 5.1.4.1.

Technique	Modeling	Analysis
Abuse Cases	Sufficient and suitable to model the security requirements of a small and medium systems. However, it might be difficult to present detailed abuse cases. Thus, for a huge system it is preferable to combine the Abuse Cases technique with the Attack Trees technique to present the detailed abuse cases.	Adequate for a general and an average level of analysis. However, for more detailed analysis the defined abuse cases can be linked to the Attack Trees technique whereas each abuse case can be set as a root node for an attack tree.
Misuse Cases	Similar to the Abuse Cases technique, but this technique covers more cases than Abuse Cases technique.	Similar to the Abuse Cases technique.
Data Sensitivity and Threat Analyses	Sufficient to produce very general patterns.	Helpful if it is used with any other technique.
Strategic Modeling	Sufficient and suitable to model the security requirements of any system. However, the modeling will be in a high level; thus it can be combined with the Attack Trees technique to model a deeper level of the security requirements. Moreover, adding descriptions for the attacker's capability is preferable.	It provides acceptable level of analysis where the analyst can control the level to be high or deep level of analysis.
Attack Trees	Sufficient and suitable to model the different ways of the attacks against the system in detail. However, it is preferable to use it with the Abuse Cases, Misuse Cases or Strategic Modeling techniques.	It provides acceptable level of analysis where the analyst can control the level to be high or deep level of analysis.

Table 6: Comparison between the level (depth) for modeling and analyzing the security requirements for a system.

6.2 FITTING SECURITY REQUIREMENTS ENGINEERING INTO CONVENTIONAL REQUIREMENTS ENGINEERING

In order to fit security requirements into the conventional requirements engineering process we have to involve the security requirements from the beginning of the software development life cycle. However, we found that using only one of the previously mentioned and discussed techniques will not be suitable enough to fit the security requirements into the conventional requirements engineering. Thus, we suggest combining some of these techniques together; therefore we found that the best combination is by using either Abuse Cases technique or Misuse Cases technique with the Attack Trees technique that may lead to a sufficient level of representing security requirements of the system under consideration. The Abuse Cases and Misuse Cases techniques are easy to understand comparing to the other techniques. On the other hand, the Attack Trees technique can be used to expand the produced abuse cases or misuse cases to reach a deep level of analysis and representation for security requirements (i.e. each abuse cases or misuse cases can be modeled as the root node of an attack tree). However, involving the main idea from the Data Sensitivity and Threat Analyses technique is highly recommended since it may lead to saved project time. Another good combination is by using the Strategic Modeling technique and the Attack Trees technique together since the Strategic Modeling technique focuses on high-level security requirements, without forcing the modeler to investigate the security mechanisms immediately, however the Attack Trees technique helps in defining and specifying the different ways of attacking the system; thereby collaboration between both of them might be considered as very effective.

6.3 SUGGESTIONS AND RECOMMENDATIONS

Security experts should be involved in the software development life cycle for many reasons such as security experts are more capable and have high level of knowledge and experience of different kind of vulnerabilities, threats, security theories and suitable thwart mechanism(s) against certain kind of vulnerabilities and/or threats. Thus, involving security experts in the development process will increase the possibility and the chance of developing a secure system; also it will increase the opportunity for the development team to learn from them, thereby increasing their knowledge, understanding and experience of how to deal with security issues within the development process.

Also, we recommend developing a standard catalogue that specifies different aspects regarding the security and its concepts in detail. However, we recommend creating and updating such catalogue by mainly involving security experts. Such catalogue may be considered helpful for the system analysts to recognize the different security areas that the system should cover.

In addition, since all of the previously mentioned techniques use trusted assumptions regarding the attacker, we suggest developing a standard method to classify attacks according to standard criteria such as attacker educational level, the approximate cost to perform a certain attack, the approximate needed time to complete a certain attack, etc. Of course, developing such classification is not an easy task and it needs time but on the other hand it will positively help in saving time during the development process especially in the analysis phase.

Developing a highly secure system is possible but it is a matter of time, budget and cooperation between different communities. Take SET (Secure Electronic Transaction) for instance which is considered as an open encryption and security specification designed to protect credit card transactions on the Internet [40] [47]. A wide range of companies from different communities were involved in the development process including IBM, Microsoft, Netscape, RSA, Terisa and Verisign [40]. The cooperation outcome was a total of 971 pages of specification and a successful project. Thus, the main lesson that can be learned from the

SET project is the importance of the cooperation and involvement of the specialist in specific domains during the software development life cycle.

Another important issue that must be kept in mind is that the software that is considered to be a secure nowadays might not be secure in the future; this depends on what technology can provide in the future. For example, an encryption algorithm with a strong key that is considered to be impossible to get broken nowadays since it needs a massive period of time to generate the key using one of the hacking tools, could be easily broken in the future if a more powerful machines are available on the market. Thus, we can conclude that the security is not a product but it is a certain kind of processes that should be updated and discussed all the time.

6.4 MISSING THINGS

Assessing the countermeasures is something that still missing in all of the previously mentioned techniques. Thus, by assessing the used countermeasure(s) to thwart a certain threat, we can answer questions such as:

- To which level the used countermeasure may be considered to be effective?
- What is the protecting percentage that this countermeasure can provide?
- What is the faulty beliefs regarding this countermeasure?
- When can the countermeasure be considered as useless?
- What are the alternative countermeasures?
- Is the selected countermeasure is the most suitable one among the others?
- What is it good for?

However, it is worth to mention that Liu et al. [16] suggest a countermeasure analysis to decide whether the impacts of the threats are reduced to an acceptable level.

Other kind of supporting tools are needed to support these techniques. These tools can help in minimizing required time to model, analyze and manage the security requirements; thereby speeding up the development process.

Chapter 7

CONCLUSIONS

The need for developing secure systems is increasing rapidly, and this is affected and caused by the increasing use of the Internet. However, software engineers recognize that there is no ideal solution to security problems; thereby they recognize the need for techniques to model and analyze security requirements to understand the security attacks, security mechanisms and security services in an adequate level. Thus, understanding the different techniques that are used to model and analyze security requirements is an important aspect in order to recognize the different techniques that are used to satisfy the security requirements of the system under investigation. Thus, various interesting techniques which exist to model and analyze security requirements have been described and investigated in this thesis; namely these techniques are: the Abuse Cases, Misuse Cases, Data Sensitivity and Threat Analyses, Strategic Modeling, and Attack Trees techniques. Each of these techniques presents some advantages and disadvantages as mentioned in Chapter 5.

After investigating the previously mentioned techniques, we conclude the following answers to our research questions:

How can security requirements be specified, modeled and analyzed? Security requirements can be specified during the elicitation phase by requirements engineer in order to model and analyze them using one of the previously mentioned techniques. Moreover, using a technique such as Attack Tree can lead to figure out more detailed security requirements by decomposing security requirements that are described in a high level as mentioned in section 4.1.

What are the differences between different techniques of modeling and analyzing security requirements? Generally, these techniques differ in the modeling process, used notation, depth of the analysis process for security requirements etc. The differences between the different techniques are mentioned in section 6.1 where we constructed different comparisons between different techniques according to different criteria.

What are the benefits to have different techniques of modeling and analyzing security requirements? These techniques can handle security requirements in different levels (e.g. Attack Trees technique provides acceptable level of analysis where the analyst can control the level to be high or deep level of analysis, while Abuse Cases and Misuse Cases Techniques provide general level of analysis). Moreover, some of them such as Abuse Cases and Strategic Modeling techniques were built on standard notations which are the UML and I* notations respectively. This helps requirements engineer to decide which technique to use. In addition, combining two techniques together can reduce the negative effects of each other and might be considered to be beneficial.

Do these techniques cover all the concepts of security? Generally, all of these techniques cover the concepts of security but in different levels. Attack Tree technique is considered to cover the security concepts in the highest level which is *Medium-high* and Data Sensitivity and Threat Analysis technique is considered to cover the security concepts in the lowest level which is *low* (see section 6.1).

Our conclusion was that using only one of the previously mentioned and discussed techniques will not be sufficient to satisfy security requirements of the system under investigation. Consequently, we consider that it would be beneficial to use one of the following two combinations:

- The first combination consists of merging the Abuse Cases technique or Misuse Cases technique with the Attack Trees technique.

- The second combination is using the Strategic Modeling technique and the Attack Trees technique together.

There are many reasons behind considering these two combinations effective and suitable to satisfy security requirements; some of them are:

- By using the Abuse Cases technique, Misuse Cases technique or Strategic Modeling technique, the requirement engineer can focus on eliciting and modeling the significant security requirements on a high and/or medium level since he/she recognizes that using Attack Trees technique will produce new security requirements that he/she did not think about.
- There are supporting tools for each of the techniques which might lead to speed up the process.
- Referring to the discussion that has been carried out in section 6.1, we can note from Table 3 that the Abuse Cases, Misuse Cases and Attack Trees techniques have a positive value for all criteria used to compare different technique. Thus, combining Abuse Cases technique or Misuse Cases technique with Attack Trees technique may help in increasing the efficiency, thus it can be considered as significant. For example, since the possibility of using the Abuse Cases, Misuse Cases, and Attack Trees techniques in practice has been considered to be high, the expected possibility for using a combination between them is considered to be high.
- The Abuse Cases and Misuse Cases techniques can be used to model security requirements in a sufficient way; at the same time Attack Trees technique provides acceptable level of analysis where the analyst can control the analysis level to be high or deep. Thus combining the Abuse Cases technique or Misuse Cases technique with Attack Trees technique considered as interesting issue.
- Using the Strategic Modeling technique and the Attack Trees technique together can help in covering the main weak point in the Strategic Modeling technique which is not specifying the attacker characteristics which is provided by Attack Trees technique.

Moreover, the concentration on using the Attack Trees technique is due to the reusability of the attack trees (i.e. reusing a previously built attack trees when necessary either in the same project or in another project); in addition using Attack Tree technique helps in covering a wide range of attacks that can cover all security concepts in a deep level of investigation. Finally, we can conclude that security is a continuous process and is not a specific product that could be bought and employed immediately in a certain system.

Chapter 8

FUTURE WORK

This thesis discussed and investigated different sorts of techniques used to model and analyzed security requirements from the academia point of view. Future work will continue in this area but in different focuses. The following subsections show some directions of future work.

8.1 FURTHER PERFORMANCE EVALUATIONS

Due to time to market constraint, the performance of any of the previously mentioned techniques is considered as a critical aspect and it plays a main role in deciding whether to use the technique or not. Therefore, in the future work it would be beneficial to evaluate the performance of these techniques in more detail in order to provide accurate results (e.g. percentage result that describes the overall performance of the technique according to specific criteria). Moreover, not only the used criteria should be discussed in greater detail, but the calculation process and steps as well.

8.2 PRACTICAL STUDY

As it was mentioned in section 3.3, using triangulation of data sources and involving the use of different ethnographic techniques like observation, interviews etc. is preferable. Thus, conducting another study to discuss and investigate these techniques from the practical point of view will be beneficial. Such study might be performed using a questionnaire survey, observations and/or interviews in order to obtain requirements engineers and security expert's opinions and thoughts regarding these techniques. Also, such study helps in recognizing the exact techniques that are used to model and analyze security requirements in practice. Therefore, investigating the carried out modifications on these techniques, if some modifications have been carried out. Moreover, such study helps in comparing these techniques according to real criteria which should be elicited from the used ethnographic techniques used in conducting the study.

8.3 MEASURING DIFFERENT COMBINATIONS

Future work will continue exploring and measuring the possibility of using the two suggested combinations mentioned in section 6.2 where we stated some benefits of these combinations. Thus, it would be interesting to know whether these techniques can be combined in practice as well as comparing the theoretical benefits with the actual benefits.

8.4 DEALING WITH SECURITY REQUIREMENTS IN OTHER REQUIREMENTS ENGINEERING PHASES

As stated in Section 4.2, the topic of the thesis covers only the first three phases of the security requirements within the requirements engineering process. Thus, it would be interesting to study the validation, management and testing phases with respect to the security requirements. By combining such study with this thesis, the reader will have a full overview regarding how we can deal with security requirements along with the requirements engineering process. Moreover, such study helps in clarifying whether the validation and testing methods that are used to deal with functional requirements are suitable to deal with security requirements.

REFERENCES

[1] Crook R., Ince D., Lin L., and Nuseibeh B., "*Security Requirements Engineering: When Anti-requirements Hit the Fan*". Requirements Engineering, IEEE, 2002, pages 203–205.

[2] Anderson R., *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, Inc., Wiley Computer Publishing, ISBN 0-471-38922-6, 2001.

[3] Devanbu P. and Stubblebine S., "*Software Engineering for Security: A Roadmap*", In Proc., 2000, Pages 227–239.

[4] Giorgini P., Massacci F., Mylopoulos J., Zannone N., "*ST-Tool: a CASE Tool for Security Requirements Engineering"*, Industrial Electronics, IEEE, 2005, pages: 451-452.

[5] Kis M., "*Information Security Antipatterns in Software Requirements Engineering*", 9th Conference of Pattern Languages of Programs (PloP), Member IEEE, 2002, Pages: 1-7.

[6] Liu L., Yu E. and Mylopoulos J., "*Analyzing Security Requirements as Relationships among Strategic Actors*", 2nd Symposium on Requirements Engineering for Information Security (SREIS'02), Raleigh, North Carolina, October 16, 2002.

[7] Chaung L., Nixon B., Yu E., and Mylopoulos J., *Non-Functional Requirements in Software Engineering,* Kluwer International Series in Software Engineering, ISBN: 0792386663, London, 1999.

[8] Irvine C., Levin T., Wilson J., Shifflett D. and Pereira B., "*An Approach to Security Requirements Engineering for a High Assurance System** ", Requirements Engineering, Springer, Vol. 7 Issue: 4, 2002, Pages: 192-206.

[9] Trochim W., Research Methods Knowledge, 2006, available at http://socialresearchmethods.net/kb/introval.htm, Last Revised: October 20, 2006, last check December 25, 2006.

[10] Sommerville I., Sawyer P., *Requirements engineering: a good practice guide*, John Wiley & Sons, IBSN 0471974447, 1998.

[11] Mylopoulos J., Chung L. and Nixon B., "*Representing using nonfunctional requirements: a process-oriented approach*", Software Engineering, IEEE, Vol.18, issue: 6, 1992, pages: 483–497.

[12] Lamsweerde v., "*Requirements engineering in the year 00: a research perspective*", Software Engineering, IEEE, 2000, pages: 5-19.

[13] Parnas D. and Madey J., "*Functional documents for computer systems*", Science of computer programming, Elsevier, Vol. 25, Issue: 1, 1995, pages: 41–61.

[14] Hester J., Yurcik W., and Campbell R., " *An Implementation-Independent Threat Model for Group Communications*", SPIE Security and Defense Conference / Program on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security (OR21) , Orlando FL USA, April 2006, Page:1- 12.

[15] Giorgini P., Massacci F., Mylopoulos J. and Zannone N., "*Modeling security requirements through ownership, permission and delegation*", Industrial Electronics, IEEE, 2005, Pages: 167-176.

[16] Liu L., Yu E. and Mylopoulos J., "*Security and privacy requirements analysis within a social setting*", Requirements Engineering Conference, IEEE, 2003, Pages: 151-161.

[17] McDermott J. and Fox C., "*Using abuse case models for security requirements analysis*", Computer Security Applications Conference, IEEE, 1999, Pages: 55-64.

[18] Sindre G., Opdahl A., "*Eliciting security requirements by misuse cases*", Technology of Object Oriented Languages and Systems, IEEE, 2000, Pages: 120-131.

[19] Lamsweerde A., Brohez S., De Landtsheer R., and Janssens D., "*From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering*", In *Proc. of RHAS'03*, pages 49–56, 2003.

[20] Gamma E., Helm R., Johnson R. and Vlissides J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[21] Cheng B., Konrad S., Campbell L., and Wassermann R., *"Using Security Patterns to Model and Analyze Security Requirements"*, Technical Report MSU-CSE-03-18, Department of Computer Science, Michigan State University, available at http://www.utdallas.edu/~hck042000/source2.pdf.

[22] Giorgini P., Massacci F., Mylopoulos J., and Zannone N., *"Filling the gap between Requirements Engineering and Public Key/Trust Management Infrastructures"*, In *Proc. Of EuroPKI'04*, *LNCS 3093*, Springer-Verlag, pages 98–111, 2004.

[23] Dawson C., *The Essence of Computing Projects - a Student's Guide*, Prentice Hall, Harlow UK, 2000, ISBN 0-13-021972-X.

[24] Mingers, J. "*Combining IS Research Methods: Towards a Pluralist Methodology*", Information Systems Research, Vol.12, Issue: 3, 2001, Pages: 240-259.

[25] Kotonya G. and Sommerville I., *Requirements Engineering Processes and Techniques*, John Wiley & Sons Ltd, England, 1998.

[26] Holtzblatt K. and Beyer H., "*Requirements Gathering: The Human Factor*", Communications of the ACM, Ebsco, Vol.38, Issue: 5, 1995, Pages: 30-33.

[27] Sommervills I., "*Integrated Requirements Engineering: a Tutorial*", IEEE Software, Ebsco, Vol. 22, Issue: 1, 2001, Pages: 16-24.

[28] Nuseibeh B. and Easterbrook S., "*Requirements Engineering: A Roadmap*", IEEE-CS, 2000, Pages: 35-46.

[29] Chris R., "*Requirements and psychology*", IEEE Software, Ebsco, Vol. 19, Issue: 3, 2002, Pages: 16-18.

[30] Hofmann H.F. and Lehner F., "*Requirements Engineering As a Success Factor In Software Projects*", IEEE Software, IEEE, Vol. 18, Issue: 4, 2001, Pages: 58-66.

[31] Goguen J.A and Linde C, "*Techniques for Requirements Elicitation",* Requirements Engineering, IEEE, 1992, Pages**:** 152-164.

[32] Paetsch F., Eberlein A. and Maurer F., "*Requirements Engineering and Agile Software Development*", Enabling Technologies, IEEE, 2003, Pages: 308-313.

[33] Westfall L., "*Software Requirements Engineering: What, Why, Who, When, and How*", Software Quality Professional, Proquest, Vol. 7, Issue4, 2005, Pages: 17-26.

[34] Kuloor C., Eberlein A., "*Aspect-oriented requirements engineering for software product lines*", Engineering of Computer-Based Systems, IEEE, 2003, Pages: 98-107.

[35] Firesmith D., *"Engineering Security Requirements"*, in Journal of Object Technology, vol. 2, no. 1, 2003, Pages 53-68.

[36] Zhang S., *"Integrating Non-Functional Properties to Architecture Specification and Analysis"*, Information Technology: New Generations, Third International Conference, IEEE, 2006, Pages: 112-117.

[37] Jürjens J., "*Using UMLsec and goal trees for secure systems development"*, Proceedings of the 2002 ACM, ACM Press, 2002, Pages 1024-1030.

[38] McGraw G., *"Misuse and Abuse Cases: Getting Past the Positive"*, Security & Privacy, IEEE, 2004, Pages: 32-34.

[39] http://www.monkey.org/~dugsong/dsniff/, last check: November 2, 2006.

[40] Stallings W., *Network Security Essentials. Application and Standards*, 2nd edition, Prentice Hall, New Jersey, ISBN: 0131202715, 2002.

[41] Sindre G. and Opdahl A., *"Templates for misuse case description"*, In Proc. of the 7th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'2001), 2001.

[42] Cockburn A., *Writing effective use cases*, 1st edition, Addison-Wesley, ISBN: 0201702258, 2000.

[43] Kulak D. and Guiney E., *Use Cases: Requirements in Context*, 2nd edition, ACM Press, 2000.

[44] Cares C., Franch X., Mayol E. and Alvarez E., *"Goal-Driven Agent-Oriented Software Processes"*, Software Engineering and Advanced Applications, 2006, Page(s):336 – 347.

[45] Mouratidis H., Giorgini P., Manso G. and Philp I., *"A Natural Extension of Tropos Methodology for Modelling Security"*, citeseer, 2002.

[46] Yu E. and Mylopoulos J., *"From E-R to "A-R" - Modelling Strategic Actor Relationships for Business Process Reengineering"*, citeseer, 1995.

[47] Giorgini P., Massacci F. and Mylopoulos J., *" Requirement Engineering meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard"*, citeseer, 2003.

[48] Yu E., *"Towards modelling and reasoning support for early-phase requirements engineering"*, Requirements Engineering, Proceedings of the Third IEEE International Symposium, IEEE, 1997, Pages: 226-235.

[49] Schneier B., *"Attack Trees"*, Dr. Dobb's Journal: Software Tools for the Professional Programmer, Ebsco, Vol. 24, Issue: 12, 1999, Pages: 21-27. Available at http://www.schneier.com/paper-attacktrees-ddj-ft.html.

[50] Mauw S. and Oostdijk M., *"Foundations of Attack Trees"*, presented at Eighth Annual International Conference on Information Security and Cryptology, LNCS, Seoul, Korea, 2005.

[51] Higuero M., Unzilla J., Jacob E., Saiz P., Aguado M. and Luengo D., *"Application of 'attack trees' in security analysis of digital contents e-commerce protocols with copyright protection"*, Security Technology, IEEE, 2005, Pages: 57-60.

[52] Google advance search, http://www.google.com/advanced_search?hl=en , last check: December 15, 2006.

[53] Advanced Scholar Search, http://scholar.google.com/advanced_scholar_search?hl=en&lr , last check: December 15, 2006.

[54] Yahoo, Advanced Web Search, http://mysearch.yahoo.com/web/advanced, last check: October 3, 2006.

[55] Ask, ask advance search, http://www.ask.com/web?q=test&qsrc=119&o=0&l=dir, last check: December15, 2006.

[56] Hofmeister C., Nord R., Soni D., *Applied software architecture*, Addison-Wesley, Addison-Wesley Longman Publishing Co., Inc., ISBN:0-201-32571-3, 1999.

[57] The OME tool, http://www.cs.toronto.edu/km/ome/, University of Toronto, CANADA. Last check: December 20, 2006.

[58] Winter, G., "*A comparative discussion of the notion of 'validity' in qualitative and quantitative research"*, The Qualitative Report, Vol. 4, (3&4). 2000, 1998, available at: http://www.nova.edu/ssss/QR/QR4-3/winter.html , last check: December 25, 2006.

[59] Golafshani N., *"Understanding Reliability and Validity in Qualitative Research "*, the Qualitative Report, Vol. 8, University of Toronto, 2003, Pages: 597-607, available at: http://www.nova.edu/ssss/QR/QR8-4/golafshani.pdf, last check: December 26, 2006.