Detecting Conflicts between Functional and Security Requirements with Secure Tropos: John Rusnak and the Allied Irish Bank

Fabio Massacci and Nicola Zannone
Dep. of Information and Communication Technology
University of Trento
{massacci,zannone}@dit.unitn.it

Abstract

The last years have seen a growing concern on the security of information systems and, consequently, a call to arms for including security aspects during the entire development process. Unfortunately, most proposals treat security in system-oriented terms and model information systems through the policies and security mechanisms they support. In contrast, attackers bypass such security measures by exploiting weaknesses of the socio-technical system as a whole. Many weaknesses are due to the presence of conflicts in functional and security requirements at organizational level. In this paper we show how the Secure Tropos requirements engineering methodology can be used to model such conflicts in a concrete case study: the fraud at Allied Irish Bank. In particular, the paper analyzes the vulnerabilities affecting the organization and information system of Allied Irish Bank and its subsidiary First Maryland Bancorp, that were exploited by a currency trader in order to fraudulently cover $700 million losses.

1 Introduction

The last years have seen a growing effort for integrating security into the system development process (Doan et al., 2004; Basin et al., 2006; McDermott and Fox, 1999; Schumacher, 2003; Sindre and Opdahl, 2005; van Lamsweerde et al., 2003). The basic idea underlying many of these proposals is to integrate security concerns into the information system through the authentica-

tion and access control mechanisms supported by the information system itself. However, this approach introduces a gap between security measures and the requirements of the organization as a whole where the information system is embedded. In other words, security mechanisms and policies are fitted into a pre-existing functional design which may not be able to accommodate them due to potential conflicts with functional requirements. For instance, several proposals (Clark and Wilson, 1987; Schaad and Moffett, 2002; Simon and Zurko, 1997) define separation of duty constraints to specify that critical tasks cannot be performed by single actors and to enforce conflicts of interest policies. However, they assume a prior knowledge of incompatible roles and conflicts, and do not analyze organizational requirements to understand why such constraints should be introduced and the effects of their introduction. Thereby, some crucial constraints can be omitted and others may affects the functionalities of the IT system.

On the contrary, RE methodologies should model and analyze organizations and their operational procedures and not just IT systems, and then derive security policies and mechanisms from the requirements analysis process. Indeed, a major source of security weaknesses and, consequently, of system vulnerabilities is the conflicts between functional and security requirements which overlap the organization and the IT system. In this setting, attackers might exploit such vulnerabilities for bypassing security mechanisms rather than break them (Anderson, 1994). As such the system designer will not spot it nor it will be detected by the manager in charge of the procedures. Thereby, the detection of conflicts in the requirements specification of the whole socio-technical system is a crucial issue in avoiding system vulnerabilities.

Internal attacks are even more harmful than external attacks since they are being performed by allegedly trusted users that can easily bypass access control mechanisms precisely because they are trusted. So, trust plays a key role in the development of secure IT systems (Giorgini et al., 2006). Trust is related to belief in honesty, trustfulness, competence, and reliability (Blomqvist and Ståhle, 2000; Castelfranchi and Falcone, 1998; McKnight and Chervany, 1996) and it is fundamental to build collaboration between humans and organizations (Axelrod, 1984). Yet, very few security engineering methodologies take into account trust aspects. This modeling absence affects the decision on the security measures imposed on the system. In particular, such measures might be excessive in some cases and inadequate in others. For instance, system designers may not introduce security measures since they may implicitly assume trust relationships among users, that are not in the domain. Alternatively, system designers may introduce expensive mechanisms for protecting a trusted system that has not been perceived as a trusted system

by the designers themselves. To solve this problem, designers should model the organizational settings in terms of social relationships among the actors involved in the system.

In previous work (Giorgini et al., 2005a; Giorgini et al., 2005c) jointly with P. Giorgini and J. Mylopoulos, we have proposed Secure Tropos, a requirements engineering methodology tailored to model both the organizational environment of a system and the system itself with respect to functional and security requirements. This framework is founded on the notions of ownership, provisioning, trust, and delegation in order to define entitlements, capabilities and responsibilities of stakeholders and system's actors and their transfer. Together with a graphical modeling framework, the authors also proposed a formal framework based on Datalog (Leone et al., 2005) that allows system designers for an automatic analysis of the system requirements.

Although the application of Secure Tropos to different case studies (e.g. (Massacci et al., 2005)) has revealed its ability to identify conflicts among functional and security requirements at the organizational level, we notice that conflicts might be concealed within the requirements specified at different levels (Giorgini et al., 2005b). Essentially, modeling and analyzing only the structure of the organization could be not sufficient to state that the system is secure. Actually, retrospectively untrusted agents can play trusted roles of the organization in order to gain personal advantage from their position in the organization itself. This shows that comparing the structure of the organization with the concrete instance of the organization (i.e., the agents that playing some roles in the organization and relations among them) is needed to bring conflicts to light.

1.1 The contribution of this paper

This paper presents an application of the Secure Tropos methodology to a real case study concerning the fraud to the detriment of Allied Irish Bank performed by John Rusnak. The aim of the paper is twofold. On one side, the paper intends to show that the Secure Tropos concepts and primitives are sufficient to capture high-level functional and security requirements. In particular, the focus is on modeling the Allied Irish Bank's policies (that should hold for every employee), the role of Rusnak in First Maryland Bancorp's organizations, and his personal relations with the other employees in the bank, based on official documents (Promontory Financial Group et al., 2003; United States Department of Justice, 2002). On the other side, the focus is on the capabilities of Secure Tropos for detecting vulnerabilities that may affect the structure of an organization and its information system. In particular, it is shown how vulnerabilities

exploited by Rusnak can be identified by comparing the structure of the organization with the concrete instance of the organization. Within this paper, we want also to emphasize the importance of detecting vulnerabilities during the early phases of the system development process. For instance, Johnson (2005) applied violation and vulnerability analysis to the same case study for understanding the root causes of security incidents. However, this analysis has been developed to assist security incident investigations so that it can be used only after the incident has taken place.

The remainder of the paper is structured as follows. In the next section we briefly present the case study. Then, we introduce Secure Tropos key functionalities (§3) and show how to model the Allied Irish Bank's organizational structure (§4) and how it has been modified after John Rusnak's hiring (§5). Then, we discuss the process for identifying potential conflicts in organizations (§6). We check for conflicts in the requirements (§7) and discuss recommended structure after the fraud (§8). Finally, we discuss related work (§9), and conclude the paper with some directions for future work (§10).

2 Case Study: John Rusnak and Allied Irish Banks

This section presents a scenario used as a running example throughout the paper. It is based on a real case involving the loss of approximately $700 million in currency transactions from Allied Irish Bank and its subsidiary First Maryland Bancorp (Promontory Financial Group et al., 2003; United States Department of Justice, 2002).

In 1983, Allied Irish Bank (AIB), the Republic of Ireland's biggest banking and financial services organization, acquired a stake in First Maryland Bancorp (Allfirst). In 1989, AIB acquired Allfirst through a merger.

In the beginning, Allfirst currency trading was run only with limited risks and a limited budget. In 1993, Allfirst recruited John Rusnak as currency trader. One reason behind his recruitment was AIB and Allfirst's desire to exploit a form of arbitrage in which Rusnak was expert. This arbitrage played on the different exchange rates between currency options[1] and currency forwards.[2] Rusnak's strategy was based on complicated deals in the foreign exchange

[1] An option is an agreement that gives the buyer the right but not the obligation to buy or sell a currency at a specified price on or before a specific future date. If it is exercised, the seller of the option must deliver the currency at the specified price.

[2] A forward is a contract that obligates the contract holders to buy or sell the currency at a specified price, at a specified quantity, and on a specified future date. These contracts cannot be transferred.

and options markets. Unfortunately, his strategy did not work and he lost a substantial amount of money. To cover the losses, Rusnak started to play with Allfirst's books and IT systems.

By exploiting weaknesses affecting Allfirst's internal procedures and IT system, Rusnak used a number of methods not only to hide losses, but also to show he was gaining money. Essentially, he created fake trades and entered them into Allfirst's books. His scheme was to simultaneously enter pairs of bogus trades into the trading system. One trade represented the sale of a currency option to an Asian bank. The other trade represented the purchase of an offsetting option from the same counterpart. Rusnak convinced and cajoled Back Office employees (where the trades are verified, as opposed to the Front Office, where the trades are made) that such trades did not need to be confirmed because no cash had actually changed hands and because they should be confirmed in the middle of the night. However, there was a significant difference between the two trades: the first option expired the day it was made, while the latter expired a month later. This scheme hid the fact that he was operating over his trading limit, which allowed him to make more trades. Moreover, these bogus options also disguised that he was taking high risks and actually losing money.

The losses were uncovered during a management review of the treasury division of Allfirst in 2001. An initial investigation at the Back Office revealed two trades supposedly made by Rusnak that had not been confirmed. Then, the supervisor of Allfirst's Back Office required an explanation from its employees that reported Rusnak's words. The supervisor didn't like this. Thus, he investigated further trades and found 12 unconfirmed deal tickets referring to recent trades with Asian banks. All unconfirmed trades were given back to Rusnak. Moreover, when the Asian banks were called, they knew nothing about such trades. The supervisor called Rusnak reporting the troubles during the confirmation of these trades. Rusnak assured the supervisor that he got confirmation of his trades. He copied the letterheads of Asian banks, typed bogus confirmations of his trades, signed them, and given them to the Back Office.

The Back Office supervisor did not like the look of the documents given by Rusnak and wanted additional confirmations by the involved Asian banks. It was Friday and the Asian markets were closed until Sunday night. Rusnak said that he would give the number of a broker who could confirm his trades to Allfirst on Sunday. Allfirst alerted the FBI after Rusnak failed to come to work on the next Monday.

Investigators considered this fraud a very complex crime for its sophisticated cover-up. If found guilty, Rusnak could be sentenced to 30 years in prison and a $1 million fine. Yet, he is

not actually charged with theft, although prosecutors said he gained nearly $500,000 in bonuses for fake bank profits (United States Department of Justice, 2002).

3 Secure Tropos

The Tropos methodology (Bresciani et al., 2004) is an agent-oriented software engineering methodology based on the i* modeling framework (Yu, 1996), tailored to describe both the system-to-be and its environment. Unfortunately, i*/Tropos lacks the ability to capture at the same time the functional and security features of the organizational setting (Giorgini et al., 2006). This issue has spurred us to enhance such a methodology. The result is the Secure Tropos methodology (Giorgini et al., 2005a; Giorgini et al., 2005c) that extends i*/Tropos with concepts suitable for modeling and analyzing security requirements together with functional requirements. However, we have focused on a modular addition so that dropping all newly proposed features makes one to return to the i*/Tropos original methodology (Giorgini et al., 2006).

Secure Tropos uses the concepts of actor, goal, task, resource and social relations for defining entitlements, capabilities and responsibilities of actors. An *actor* is an active entity that performs actions to achieve goals. Actors can be decomposed into sub-units for modeling the internal structure of the actor itself while preserving the intentional abstraction of the actor itself. Complex social actors can be modeled through three types of sub-units: agents, roles, and positions. An *agent* is an actor with concrete and physical manifestations, normally an individual person or a concrete piece of running software. A *role* is an abstract characterization of the behavior of a social actor with respect to a specific domain. A *position* represents a set of roles played by an agent. In the remainder, an agent is said to occupy a position, while a position to cover a role. A *goal* represents actors' strategic interests. A *task* specifies a particular course of actions that should be executed in order to satisfy a goal. A *resource* represents a physical or an informational entity. For sake of simplicity, the notion of service is used to refer to a goal, task, or resource. Figure 1 shows the graphical representation of the above concepts. Actors are represented as circles; goals, tasks and resources are respectively represented as ovals, hexagons and rectangles.

Secure Tropos introduces social relations, namely *objectives*, *ownership*, and *provisioning* for defining desires, entitlements, and capabilities of actors. Objectives of an actor are classic feature of a goal-oriented methodology and we will not discuss them further. The basic idea of ownership is that the owner of a service has full authority concerning access and disposition of his service.

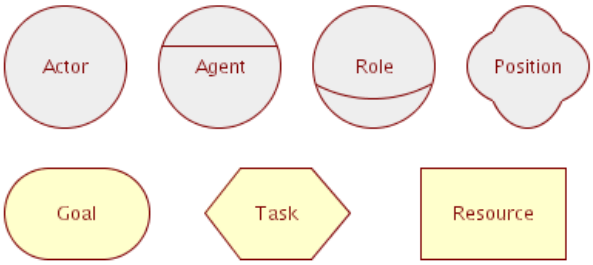

Figure 1: Graphical Representation of Secure Tropos concepts

In contrast, provisioning marks the actors who have the capabilities to deliver a service.

Moreover, Secure Tropos refines the notion of dependency offered by i*/Tropos by introducing the notion of *(dis)trust* and *delegation* (Giorgini et al., 2005b). These new social relations are used to model the transfer of entitlements and responsibilities between actors. In particular, delegation marks a formal passage of authority or responsibilities from one actor (the delegator) to another (the delegatee) in the domain of analysis. The i*/Tropos methodology has been designed with cooperative systems in mind so that a dependency between two actors means that the dependee takes the responsibility to achieve the depender's goal and he is also authorized to achieve it (Giorgini et al., 2006). The application of Secure Tropos to comprehensive case studies (e.g., (Massacci et al., 2005)) has revealed that distinguish between relations involving permission and relations involving execution is essential to verify the consistency among functional and security requirements. To this extent, we distinguish the notions of delegation of permission and delegation of execution. *Delegation of permission* is used when in the domain of analysis there is a formal passage of authority, that is, the delegator authorizes the delegatee to access a resource, execute a task, or achieve a goal. In contrast, a *delegation of execution* is used to model a delegation of responsibilities, that is, to model situations where the delegator wants the delegatee to deliver a service since he has not the capability to deliver it by himself.

Example 1 *Allfirst treasury office, who needs foreign exchange rates,* delegate *the* execution *of the task of providing them to Reuters, a global information company. On the other hand, Allfirst treasury office, who is the owner of the currency trading activities,* delegates *the* permission *to enter trades in the Front Office and settle trades in the Back Office.*

System designers might need to model systems where some actors must delegate services to other actors they do not trust. Thus, it is convenient to separate the concept of trust from the concept of delegation. Essentially, trust is a relation between two actors representing the

expectation of one actor (the trustor) about the capabilities and behavior of the other actor (the trustee) (Castelfranchi and Falcone, 1998). Also in this case it is convenient to distinguishing two notions of trust: trust of permission and trust of execution. The meaning of *trust of permission* is that an actor trusts that another actor will not misuse the permission on the service. The meaning of *trust of execution* is that an actor trusts that another actor has the direct or indirect capability to deliver the service.

Many domains also demand the possibility to make explicit negative authorizations to help the designer in shaping the perimeter of positive authorizations. For instance, in distributed systems, an actor possessing the right to use the service, can delegate the authorization on that service to the wrong actor. In this setting, it is not always possible to deny an actor to access a particular service. Thus, we propose an explicit distrust relationship as an approach for handling this type of scenario. Obviously, there are various reasons of distrusting in agents such as unreliability and abuse, but their analysis goes well beyond the scope of this chapter. As done for trust, the notion of distrust of permission is separated from the notion distrust of execution. The occurrence of *distrust of permission* in the model means that an actor believes that another actor may misuse a service, and the presence of *distrust of execution* means that an actor believes that another actor may have not the capability to deliver a service.

Different modeling activities contribute to the acquisition of a first requirements model, to its refinement into subsequent models:

Actor modeling, which consists of identifying and analyzing both the actors of the environment and the system's actors. Furthermore, this activity identifies actors which own services and actors which have the capability to provide services beside the identification of their objectives.

Trust modeling, which consists of identifying the trust and distrust (both of execution and permission) relations among actors involved in the system.

Delegation (of execution) modeling, which consists of identifying actors which delegate to other actors the execution of services.

Delegation (of permission) modeling, which consists of identifying actors which delegate to other actors the permission on services.

Goal refinement, which consists of refining requirements. This activity is conducted from the perspective of single actors through AND/OR decomposition.

These modeling activities correspond to different kinds of diagrams: *actor diagram*, *trust model*, *functional requirements model*, and *trust management implementation*. In particular, the actor diagram represents the actors involved in the system along with their desires, entitlements, and capabilities; the trust model, functional requirements model, and trust management implementation enrich the actor diagram by representing the trust network, delegation of execution network and delegation of permission network, respectively. Such diagrams are then refined in the goal refinement activity.

4 Applying Secure Tropos to the Case Study

This section presents an application of the modeling phases to the running example. First, the actors of the environment and system's actors are identified with their goals. Then, the structure of the organization of AIB and Allfirst is analyzed by modeling the social relations among actors.

4.1 Modeling Actors

The first activity in the requirements analysis process is actor modeling. Below some of the stakeholders belonging to the running example are listed.

Allied Irish Bank (AIB) is the Republic of Ireland's biggest banking and financial services organization. After merging with Allfirst, AIB allowed Allfirst a large amount of local autonomy. Allfirst continued to have its own management team and board of directors. However, AIB wanted to control Allfirst operations, and so it appointed one of its senior managers as Allfirst Treasurer.

First Maryland Bancorp (Allfirst) is an AIB subsidiary. For sake of simplicity, we take into account only Allfirst Treasury department. This department is managed by Allfirst Treasurer. Allfirst's treasury operations are divided into three areas and each of them is managed by a specialized office:

Front Office, which is responsible for treasury found management. This office is manage by the Treasury Funds Manager and includes the Foreign Exchange Trading Office where Currency Traders work.

Middle Office, which is responsible for liability and risk management. This office includes the Risk Control Group that was responsible for risk control and analysis.

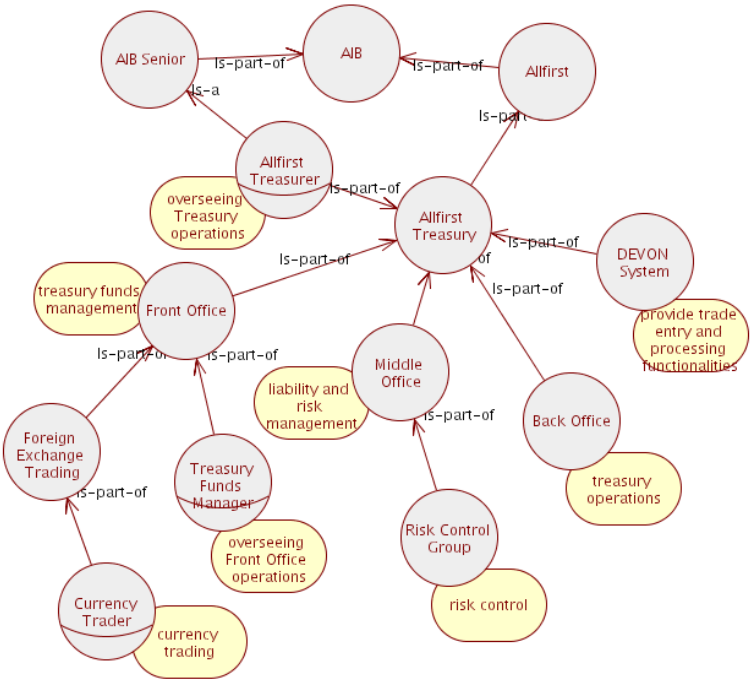

Figure 2: AIB and Allfirst's Organization

Back Office, which is responsible for treasury operations.

Last, but not least, a component of Allfirst is the DEVON System, Allfirst's information system. It was designed for providing trade entries and processing functionalities.

Foreign Trading Office represents the Treasury of banks with which Allfirst makes deals.

Reuters is a global information company providing information tailored for professionals in the financial services, media and corporate markets.

Figure 2 shows the output of the actor modeling phase. In particular, the picture illustrates the structure of the organization of AIB and Allfirst and the responsibilities of each actor.

4.2 Modeling Trust and Delegation

The requirements modeling process proceeds introducing the social relations among actors involved in the system and the consequent integration of security and functional requirements.

Figure 3 shows the relations among the Front Office and the Middle Office, and the other actors of the system. In the picture, ownership relations are represented as edges between an actor and a service labeled by **O**. Labels **Te** and **Tp** indicate trust of execution and trust of

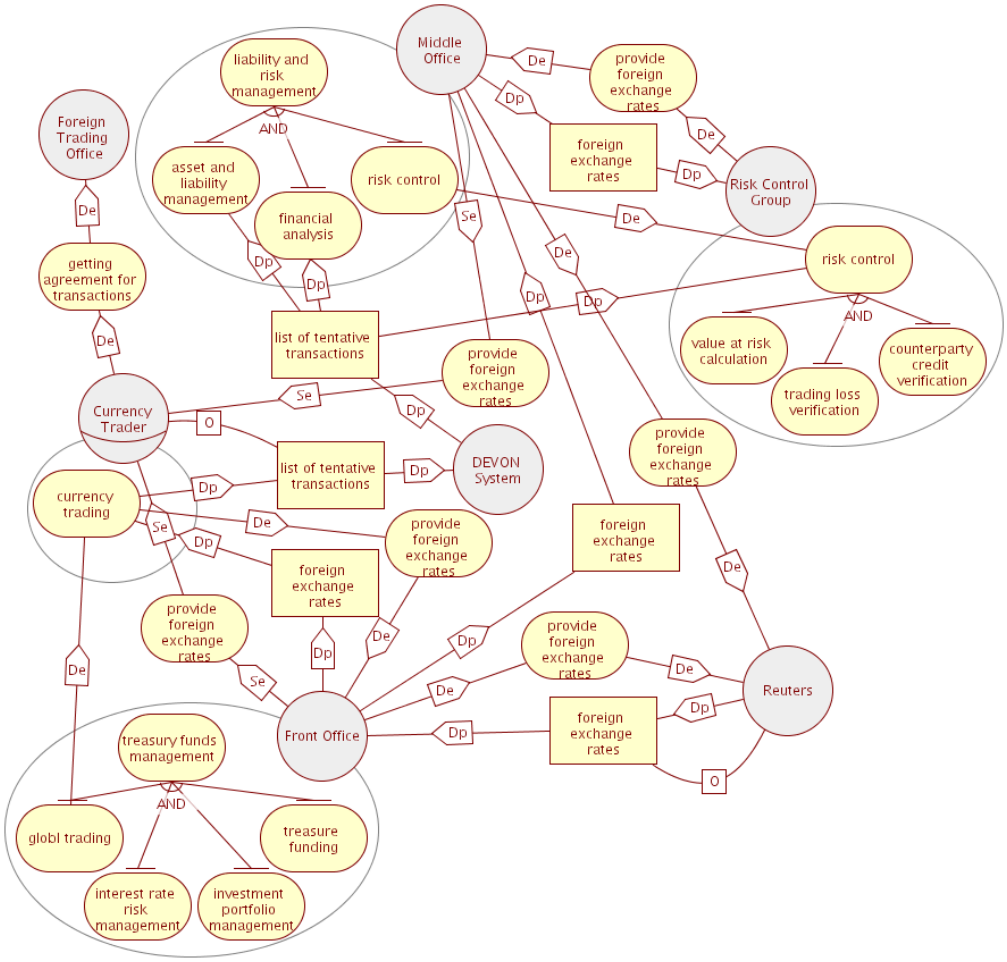

Figure 3: Front and Middle Offices' Organizational Structure

permission relations, respectively. Then, label **Dp** is used to model the actual transfer of rights, and **De** to model delegation of execution. Finally, distrust of execution and distrust of permission relations are represented through edges labeled by **Se** and **Sp**, respectively.

- The Front Office was responsible for treasury funds management. This task was decomposed into four main operations, namely treasure funding, interest rate risk management, investment portfolio management, and global trading (Promontory Financial Group et al., 2003, pag. 6).

- The Currency Trader was appointed by the Treasury Found Manager to perform currency trading operations, a particular kind of global trading operations (Promontory Financial Group et al., 2003, pag. 6).

- The Currency Trader negotiated currency options and currency forwards with the Foreign Trading Office. Once the trader reached an agreement with the counterpart, he entered information about transactions into the DEVON System (United States Department of Justice, 2002, pag. 4).

- The Middle Office was responsible for asset and liability management, financial analysis, and risk control (Promontory Financial Group et al., 2003, pag. 6).

- The Risk Control Group was appointed by the Middle Office Manager to perform risk control, and in particular for Value at Risk[3] computation, trading loss verification, and counterpart credit verification (Promontory Financial Group et al., 2003, pag. 6).

- Employees in Treasury offices needed foreign exchange rates for performing their duties and required this information from their own directors (Promontory Financial Group et al., 2003, pag. 16-18).

- Treasury office directors requested foreign exchange rates from Reuters (Promontory Financial Group et al., 2003, pag. 16).

- However, Allfirst did not want to pay an additional fee (nearly $10,000 for each office). Thus, it decided to download foreign exchange rates from Reuters onto the front office's server and then to copy them into the machines of other offices (Promontory Financial Group et al., 2003, pag. 16).

- Each employee of Treasury offices then accessed the information system of the office in which he was employed for getting foreign exchange rates.

- Notice that Allfirst's fund management policies required that Treasury offices performed their duties using prices obtained from sources independent of currency traders (Promontory Financial Group et al., 2003, pag. 16). These policies imply the presence of a distrust relation between Treasury offices and currency traders for providing foreign exchange rates.

- The Risk Control Group accessed the DEVON System in order to get information about transactions made by traders for performing its duties and in particular for computing Value at Risk (United States Department of Justice, 2002, pag. 5).

We now analyze the relations between the Back Office and the other actors.

[3]See (Jorion, 2000) for more details on Value at Risk.

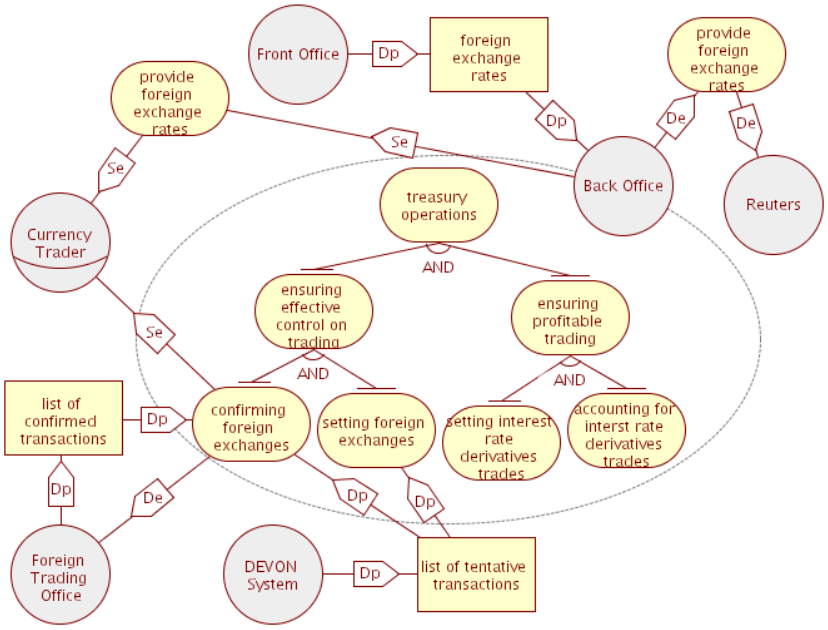

Figure 4: Back Office's Organizational Structure

- The Back Office was responsible for ensuring effective controls on trading and ensuring profitable trading. These operations was decomposed into sub goals and, in particular, ensuring effective controls on trading was decomposed into confirming foreign exchanges and setting foreign exchanges, and ensuring profitable trading into setting interest rate derivatives trade and accounting for interest rate derivatives trade (Promontory Financial Group et al., 2003, pag. 6).

- To achieve its duties, the Back Office needs some information. In particular, it requires the list of tentative transactions for confirming foreign exchanges and setting foreign exchanges. Thus, the Back Office accessed the DEVON System in order to get information about transactions made by traders for confirming them (United States Department of Justice, 2002, pag. 4).

- The Back Office contacted the Foreign Trading Office in order to confirm transactions (United States Department of Justice, 2002, pag. 4-5).

- Allfirst Treasury policies required that all trades must be confirmed by the Back Office (Promontory Financial Group et al., 2003, pag. 15). In other words, the company (and so the Back Office) distrusts a Currency Trader to confirm his own transactions.

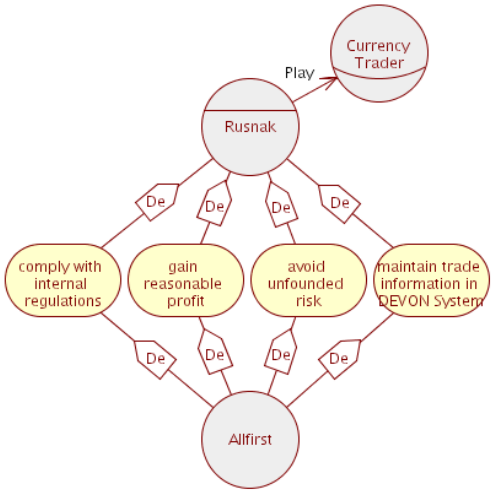

Figure 5: Rusnak's Recruitment Obligations

A graphical representation of above requirements is given in Figure 4.

5 Capturing the Position of Rusnak with Secure Tropos

This section presents the changes in Allfirst's organization after Rusnak's hiring. Figure 5 presents the obligations that Rusnak took in charge when he was employed.

- Rusnak was employed as Currency Trader by Allfirst (Promontory Financial Group et al., 2003, pag. 7).
- Rusnak was obligated to comply with internal bank regulations and procedures in performing his duties (United States Department of Justice, 2002, pag. 1).
- Rusnak was required by Allfirst to perform his duties in a way that should gain reasonable profit for the bank itself without incurring unfounded risk (United States Department of Justice, 2002, pag. 2).
- Rusnak was obligated to maintain accurate information about his trading activities in the bank information system (United States Department of Justice, 2002, pag. 2, 5).

Figure 6 presents Allfirst's organizational structure with Rusnak.

- Rusnak convinced employees in the Back Office to accept his own confirmations and not to confirm some of his transactions by arguing that his counterparts were Asian banks and

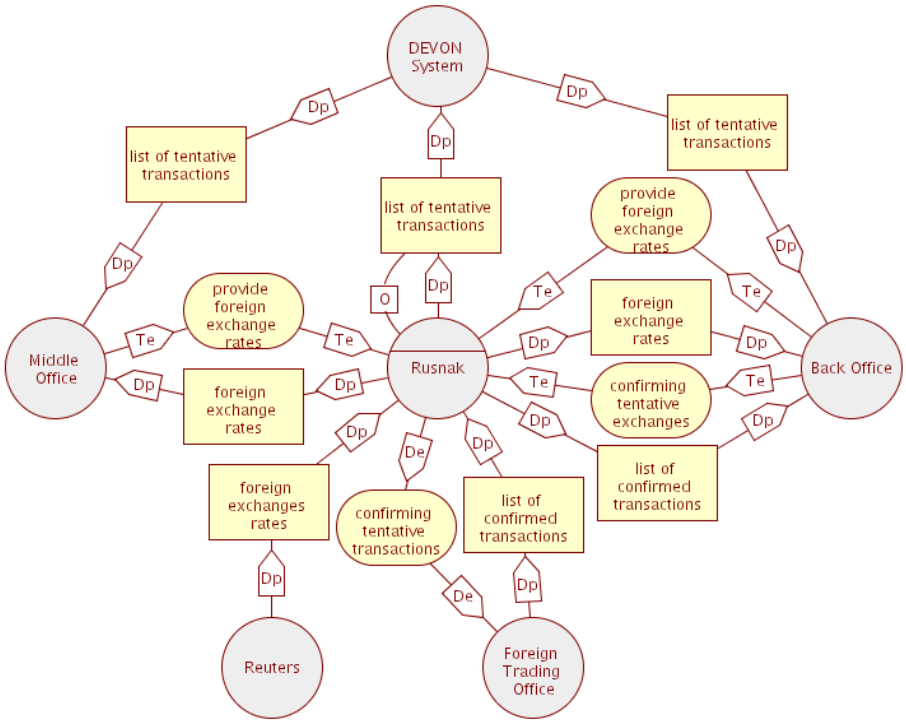

Figure 6: Allfirst's Organization with Rusnak

employees must get up in the middle of the night in order to perform their duties and by claiming that certain transactions did not require confirmation (Promontory Financial Group et al., 2003, pag. 11, 15) (United States Department of Justice, 2002, pag. 7). We summarize these statements with a trust relation between Back Office employees and Rusnak for confirming foreign exchanges.

- Allfirst did not want to pay for a dedicated Reuters feed to the Back Office. Rusnak argued that he needed to continually monitor foreign exchange rates for checking his value at risk. Thus, Allfirst analyst designed an architecture in which the Reuters feed was directly downloaded onto Rusnak's machine. Then, Treasury offices got data from Rusnak's machine (Promontory Financial Group et al., 2003, pag. 16). This scenario implies a trust relation between Treasury offices and Rusnak for providing exchanges rates.

6 Modeling and Detecting Conflicts

A critical phase of the system development process is the analysis of requirements in order to detect the presence of conflicts (van Lamsweerde et al., 1998). More than often attackers exploit vulnerabilities arising from conflicting requirements, rather than break security mechanisms themselves. We will show that this is also the case here.

A number of researchers have classified conflicts among system requirements and have proposed solutions to mitigate them (Moffett and Sloman, 1994; Simon and Zurko, 1997; van Lamsweerde et al., 1998; Lupu and Sloman, 1999; Nyanchama and Osborn, 1999). Among them, Moffett et al. (1994) and Lupu et al. (1999) have recognized two main classes of conflicts: *modality conflicts* and *conflicts of goals*. Modality conflicts are defined as those conflicts that can be identified without a knowledge of the domain under analysis and includes conflicts among authorizations, among obligations, and among authorizations and obligations. Conflicts of goals are defined as specific domain conflicts and includes conflicts of duties and conflicts of interest.

According to such a classification, Secure Tropos supports requirements engineers in detecting modality conflicts (Giorgini et al., 2005a; Giorgini et al., 2005c). Essentially, this framework includes a set of properties in form of security patterns where the failure of such properties corresponds to the presence of conflicting requirements and, consequently, the presence of vulnerabilities in the system. Avoiding or mitigating such vulnerabilities may require either to modify the structure of the organization or introduce security mechanisms during the architectural design phase.

However, this analysis is not sufficient for detecting all possible conflicts. In fact, we have also recognized the importance of comparing the structure of the organization with the concrete instance of the organization. This is crucial for capturing security requirements in a domain where a trusted role can be played by an untrusted agent and vice versa. Thus, Secure Tropos has been designed in order to support two different levels of analysis (Giorgini et al., 2005b): social and individual. Social level analysis addresses the modeling of the structure of organizations by analyzing roles and positions of the organization. On the other hand, at the individual level the focus is on single agents that are defined with their entitlements, objectives and responsibilities together with the roles they play. This approach is based on the role-based access control model (Sandhu et al., 1996) and takes advantage from specifying entitlements, objectives and responsibilities into two steps: assignment of entitlements, objectives and responsibilities to roles, and assignment of agents to roles. For instance, when new resources are entered into the system,

the administrator need only to decide which roles are entitled to access those resources. Then, all agents that play those roles inherit their properties. This means that actors' entitlements, objectives and responsibilities propagate from social level to individual level. In order to cope with these issues, we refined the requirements analysis by defining the following verification process:

1. design models at both social level and individual level, independently;
2. verify consistency of models at the social level;
3. map models at the social level into models at the individual level;
4. verify consistency of models at the individual level.

Here we have a challenge. Although visual modeling has been recognized as one of the relevant aspects in Software Engineering to ease the understanding of requirements, graphical models cannot be used for an accurate requirements verification. How do we know that the actual, concrete instance of Allfirst organization does not present loopholes that Rusnak could use?

This concern has motivated the definition of a formal framework supporting the Secure Tropos methodology based on Datalog (Leone et al., 2005). Essentially, the primitive Secure Tropos concepts and relationships are modeled through Datalog predicates (Giorgini et al., 2005a; Giorgini et al., 2005c). Unfortunately, the intuitive description of the system is usually incomplete and cannot be used to perform a correct analysis. Therefore, Secure Tropos distinguishes two main types of predicates: intensional and extensional. Intuitively, extensional predicates correspond to edges and nodes of the graphical model defined by the system designer, while intensional predicates are specified by a security expert and derived by the reasoning system. Once the designer has drawn up the model (i.e., the extensional predicates), the comprehensive description of the system (i.e., the intensional predicates) is derived by using axioms.[4] Essentially, axioms are used to make explicit that information that is necessary for an accurate requirements verification. For instance, they map the social level into the individual level, complete the trust network, identify actors entitled to access a resource, execute a task, or achieve a goal, and actors confident that their objectives will be achieved.

The last phase of the requirements analysis process is the requirements verification. To this end, Secure Tropos supports such a phase through the use of formal properties. Essentially,

[4]See (Giorgini et al., 2005a; Giorgini et al., 2005c) for a complete list of axioms.

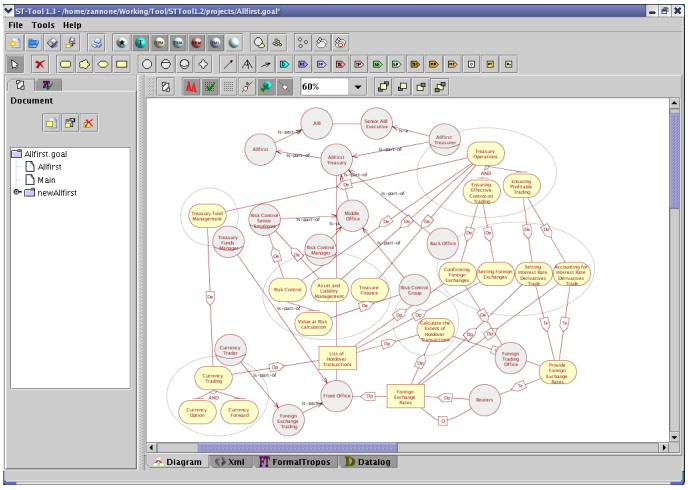

Figure 7: ST-Tool

formal properties corresponding to specific security patterns are checked in order to verify the consistency of system requirements. If all properties are not simultaneously satisfied, the system is not secure. Secure Tropos assists requirements engineers in the verification of availability, authorization and privacy requirements (Giorgini et al., 2005a; Giorgini et al., 2005c) and in the detection of trust conflicts (Giorgini et al., 2005b). In particular, the last batch of properties identifies situations where both a trust and a distrust relation exist between two actors for the same service, even when such relations are specified at different levels of analysis.

All requirements analysis phases underlying the Secure Tropos methodology are supported by a CASE tool called ST-Tool.[5] This tool is composed of two components: the ST-Tool kernel and one or more external solvers. The main component of the ST-Tool kernel is the graphical user interface (Figure 7) through which all its components are managed. This component allows designers to edit Secure Tropos models as graphs where nodes are actors and services, and arcs are relationships. Further, it aims to manage graphical objects. For instance, it supports the goal refinement phase by associating a goal diagram with each actor. A second main component of the ST-Tool kernel provides support for automatic transformations from Secure Tropos graphical models into Datalog and answer set programming specifications (Leone et al., 2005) for formal analysis. Moreover, the tool provides requirements engineers with a Datalog front-end that interacts with external solvers for an automatic requirements verification.

[5] http://sesa.dit.unitn.it/sttool/

7 Tool Supported Conflict Analysis

The fraud designed by Rusnak exploited different weaknesses affecting Allfirst's organizational structure and its information system. According to US attorney Thomas M. DiBiagio, Rusnak was motivated by different intents (United States Department of Justice, 2002, pag. 7):

- confirm his position in Allfirst by creating the sensation to gain profit for Allfirst itself;
- not to lose his employment because of huge losses;
- increase his salary with bonuses due to alleged profits.

Johnson (2005) have applied violation and vulnerability analysis to represent and modeling the output of the investigation into the causes underlying Rusnak's fraud. In particular, Johnson recognized the failures in audit and control mechanisms, individual violations, and the missed opportunities to detect Rusnak's fraud. However, this kind of analysis has been developed to assist security incident investigation so that it can be applied only once security violations have taken place.

On the contrary, our requirements analysis framework results independent from the effective occurrence of security violations. We propose to analyze the Allfirst's organizational structure and the position of Rusnak within it to detect security vulnerabilities during the system development process. Such vulnerabilities may be later exploited by a malicious actor. Notice that the comparison of requirements specified at individual and social levels does not mean that the analysis can be applied afterward. For instance, employees in the Back Office should notify their manager about the failure to confirm Rusnak's trades independently from the discovery of bogus trades. Moreover, this failure does not prove the misdoing of Rusnak. Therefore, our approach could allow system administrators to prevent attacks to the system by detecting its vulnerabilities.

The reminder of this section provides an overview of the requirements verification process through Secure Tropos. In particular, we show how the Secure Tropos methodology can detect the vulnerabilities exploited by Rusnak.

7.1 Foreign Exchange Rates

A vulnerability was based on the lack of protection of the integrity of foreign exchange rates. Conceptually, the policy on foreign exchange rates defined by Allfirst was correct: "foreign

currency rates are obtained independent of trading desk" (Promontory Financial Group et al., 2003, pag. 16). However, Allfirst did not want to pay an additional fee to Reuters for a dedicated connection for each of its offices. Furthermore, Rusnak argued that he needed such information in real time in order to continually monitoring the value at risk of his trades. Thus, Allfirst decided to develop an architecture in which rates were download on Rusnak's machine, and then the treasury offices got information from there.

Although Allfirst's analysts noticed that "this is a failed procedure" and "technically, the trader/s could manipulate the rates" (Promontory Financial Group et al., 2003, pag. 16), they did not alert Middle Office and Back Office's managers about their worries. Furthermore, this procedure has downgraded the "control market risk" rating from "good" to "weak". However, the "quality of risk management" rating (that includes the previous rating) has been only downgraded to "acceptable" (Promontory Financial Group et al., 2003, pag. 17) so that no measure was adopted in order to solve this situation.

As Allfirst's analysts have guessed, the rates spreadsheet was corrupt (Promontory Financial Group et al., 2003, pag. 17). Essentially, Rusnak has manipulated the price associate with yen and dollar in order to bypass the loss limit imposed to him by Allfirst.

By looking at the models at the social level and at the individual level it is possible identify the inconsistency between Allfirst's policies and the concrete instance of the organization. In particular, we can see the presence of a distrust relation between treasury offices and currency traders for providing foreign exchange rates at the social level (Figure 3 and Figure 4), and the presence of a trust relation between treasury offices and Rusnak for providing foreign exchange rates at the individual level (Figure 6).

Even if this conflict is "visible", it could be disregarded by the system designer due to its nature (i.e., it arises from the comparison of different levels of analysis) and the size of the requirements model. The formal framework and the supporting tool aid the system designer in automatically detecting such a conflict. As presented in Section 6, the Secure Tropos requirements verification process includes the mapping of the model at the social level into the model at the individual level, and so the distrust relation involving currency trader is inherited by Rusnak. This allows the reasoning system to easily detect the inconsistency.

7.2 Confirmation

The failure of confirmation of bogus options was one of the fundamental vulnerabilities exploited by Rusnak. Allfirst defined a policy stating that every trade made by currency traders must be confirmed by the Back Office. However, this policy was not implemented properly. Rusnak persuaded some Back Office employees to not confirm all his transactions due to both the absence of cash transfer and the time zone difference with his Asian alleged counterparts.

Bogus options were used by Rusnak for distorting the value at risk calculation. In particular, they mitigated the risk associated with his real trades. Furthermore, Rusnak needed to find a huge amount of cash to cover the losses of his real trading. Thus, he sold "deep-in-the-money" options[6] to different counterparts (Promontory Financial Group et al., 2003, pag. 11) (United States Department of Justice, 2002, pag. 9). Rusnak then inserted into the trading system fake options showing that the original option has been repurchased in order to cover his mounting losses.

Once again, this vulnerability arises from the conflicts between Allfirst's policies and the concrete instance of the organization. In particular, the Back Office distrusts currency traders for confirming transactions (Figure 4), while the Back Office employees trust Rusnak for the same goal (Figure 6). Such a conflict can be automatically detected by the ST-Tool. By mapping the social level into the individual level, the distrust relation involving currency trader is inherited by Rusnak. Thus, the tool reveals that Rusnak is at the same time trusted and distrusted for confirming transaction by the Back Office employees.

7.3 Middle Office vs Back Office

The lack of interaction between the Middle Office and the Back Office was part of the set of vulnerabilities exploited by Rusnak. The Middle Office computed the value at risk on tentative trades instead of considering trades confirmed by the Back Office. Thus, Rusnak was able to tamper value at risk by introducing bogus options in the list of tentative transactions.

The analysis does not reveal this vulnerability since it is conceptually embedded into the system requirements. However, we have not modeled the code of practice for financial market (Association Cambiste Internationale, 2005) and the trust/distrust relations that are proper to this model. Therefore, it might be that this loophole could have been captured by modeling the

[6] A "deep-in-the-money" option is an option with a price that is significantly below the market price and had large premium.

proper path of delegation of execution and trust and the corresponding mismatch of Allfirst's organizational structure.

However, by looking at the models in Figure 3 and Figure 4 it is evident the lack of relations between the two treasury offices. In particular, Figure 6 clearly shows that Rusnak controlled the information that the two offices used to perform their duties.

8 Updated Model

The resolution and mitigation of vulnerabilities is a necessary condition for successful development of secure software systems (van Lamsweerde et al., 1998). In the reminder of this section, we use Secure Tropos to illustrate and validate the solutions adopted by AIB.

After the fraud was discovered, AIB appointed Promontory Financial Group for an independent review of its internal control and risk management system. Promontory Financial Group together with the law firm of Wachtell, Lipton, Rosen and Kats analyzed AIB and Allfirst's organization and identified numerous deficiencies in the control structures at Allfirst. The vulnerabilities that we have shown in the previous section and other vulnerabilities together with some suggestions for coping with them were explained in the "Ludwig report"[7] (Promontory Financial Group et al., 2003).

Promontory Financial Group's suggestions mainly focus on the attention that AIB and Allfirst should take on policies and procedures. In particular, reviewers revealed that treasury staff was enough not expert to fulfill its duties. Thus, Promontory Financial Group (2003) suggested to retrain the current staff or to replace it with skilled personnel. This confirms the importance of analyzing the concrete instance of a organization together with its structure. Additional support to this thesis comes from recent studies (Ponemon, 2003) that reveal that information security administrators' biggest worry is employee negligence and abuse.

Promontory Financial Group (2003) pointed out a single suggestion concerning Allfirst's organizational structure: the distribution of foreign exchange rates. Following the "Ludwig report", Allfirst decided to pay for a dedicated Reuters feed to the Middle and Back Offices in order to decrease fraud risk. Figure 8 presents the new configuration for distributing foreign exchange rates among treasury offices. We argue that this is not the only solution in order to guarantee the integrity of foreign exchange rates. For instance, Allfirst could require from

[7]The Promontory Financial Group's report was called "Ludwig report" from Eugene Ludwig, the former currency controller who has written the report.

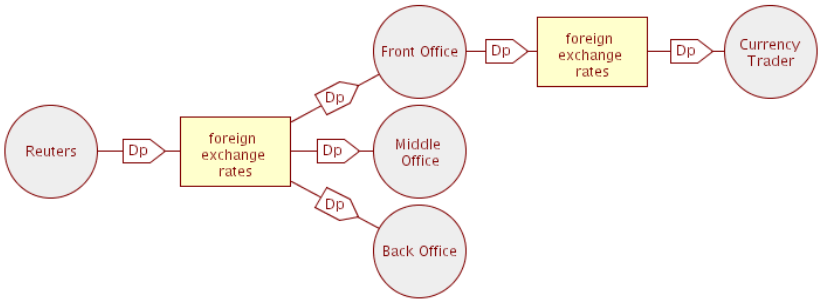

Figure 8: Distribution of foreign exchange rates

Reuters a message authentication code (MAC) together the data themselves. Then, treasury offices have to check the integrity of data before using them. This solution could allow Allfirst to avoid a dedicated connection between Reuters and every treasury office and consequently to save money.

9 Related Work

Several research efforts have addressed the issue of integrating security into the system development process. They focus on very different aspects, from design of access control mechanisms to modeling the behavior a system should avoid, from the definition of principles for conflict analysis and classification to the definition of security patterns.

Access control plays a key role in the development of secure IT systems. Proposals for specifying and enforcing access control policies can be classified under three main classes: discretionary access control (DAC) (Downs et al., 1985), mandatory access control (MAC) (Bell and LaPadula, 1976), and role based access control (RBAC) (Sandhu et al., 1996). However, these proposals focus on the specification of the access control policies and related constraints (e.g., separation of duty constraints) supported by the IT system itself and do not supports designers in the system development process. To reduce the gap between security models and system design, several research efforts have attempted to integrate access control models into Software Engineering by using or enhancing UML constructs (Doan et al., 2004; Basin et al., 2006; Ray et al., 2004; Shin and Ahn, 2000). Accordingly, these proposals manly focus on conflicts of duties (i.e., violations of separation of duty constraints). In particular, they define specific domain constraints that will be checked statically or dynamically. Doan et al. (2004) propose to incorporate the MAC model into UML diagrams. Shin et al. (Shin and Ahn, 2000) reduce the gap between

security models and system developments by proposing conceptual models for RBAC in UML. Constraints are represented as classes in class diagrams and verified by RBACController in collaboration diagrams. Basin et al. (2006) proposed SecureUML, a modeling language designed to integrate RBAC policies into a model-driven software development process. To model conflicts of duties, they introduce conflict of duty classes and then statically assign objects to subjects and subjects to conflict of duties classes through subject-role and user-group assignment. Ray et al. (2004) integrate the RBAC model in UML as patterns using diagram templates, and express RBAC constraints through the Object Constraint Language. Yet, these proposals focus only on the system-to-be and the access control mechanisms supported by the system itself, and do not analyze the organizational setting where the system-to-be will operate. This makes it difficult to understand why policies and constraints should be introduced in the design and the effects of their introduction. Moreover, they require a prior knowledge of possible conflicts, whereas our work focuses on detecting conflicts from the requirements analysis process.

Other approaches have been proposed to explicitly model behaviors that the system should avoid (McDermott and Fox, 1999; Sindre and Opdahl, 2005; van Lamsweerde et al., 2003). McDermott and Fox (1999) define abuse cases as interactions between a system and one or more actors, where the results of such interactions are harmful to the system, or one of the stakeholders of the system. Guttorm and Opdahl (2005) define misuse cases, the inverse of UML use cases, which describe functions that the system should not allow. This approach is also adopted by van Lamsweerde et al. (2003) that extend the KAOS methodology (Dardenne et al., 1993) by introducing the notion of anti-goals as the objectives of attackers.

Regarding the conflict analysis, some guidelines providing support for detecting and mitigating conflicts among requirements and policies are emerging (Lupu and Sloman, 1999; Bandara et al., 2003; van Lamsweerde et al., 1998). Lupu et al. (1999) propose to use policies overlapping techniques to detect conflicts among policies. Bandera et al. (2003) define a formal framework based on Event Calculus in order to support this approach. Essentially, they represent conflicts by constraints on events, and then simulate the system behavior through sequences of events and detect policy inconsistencies by identifying the situations in which conflicts occur. We differ from this approach since we aim at understanding why a system should comply with such constraints. van Lamsweerde et al. (1998) proposed formal techniques for detecting conflicting formulations of goals and requirements among different stakeholder viewpoints. In particular, they suggested various techniques for systematically resolving conflicts by introducing new goals

or transforming the specification of goals towards models that are not affected by conflicts. Our approach extends this work along two directions: firstly, we consider both entitlements and objectives rather than only objectives; secondly, we also detect conflicts due to conflicting social relations among actors.

Security engineering with patterns is recently becoming a hot topic of research (Cheng et al., 2003; Priebe et al., 2004; Schumacher, 2003; Schumacher et al., 2005; The Open Group, 2004). Security patterns have been proposed in order to assist in identifying and formulating security measures that are relevant to the system development. They provide ad-hoc solutions in a systematic and structured manner. Essentially, security patterns are security best practices presented in a template format. This format aids designers in identifying and understanding security concerns, and in implementing appropriate security measures even if they are not security experts (Schumacher, 2003). Currently, many efforts are addressed to the definition of a template for security patterns that is tailored to integrate security and systems engineering (Cheng et al., 2003; Schumacher, 2003; The Open Group, 2004). In particular, many solutions propose to use UML to represent structural and behavioral aspects of design.

10 Conclusion

The last years have seen a major interest for software engineering methodologies that could capture security concerns. We have proposed Secure Tropos, a methodology tailored to deal with trust and security requirements from the very early stage of design. We have shown over a complex case study the effectiveness of Secure Tropos in order to detect modality conflicts among high-level functional and security requirements. The next step is to provide automatic mechanisms also for detecting conflicts between goals such as conflicts of duties and conflicts of interests.

A more ambitious objective now is to move towards architectural design. After a preliminary analysis, we have recognized the potential of security patterns approaches (Schumacher, 2003; Schumacher et al., 2005) for dealing with this issue. Thus, we are currently defining a security pattern repository based on Secure Tropos and general schemes for representing structural and behavioral aspects of design into UML-based frameworks for security (Basin et al., 2006; McDermott and Fox, 1999; Sindre and Opdahl, 2005).

Acknowledgments

This work was partly supported by the projects RBNE0195K5 FIRB-ASTRO, 016004 IST-FP6-FET-IP-SENSORIA, 27587 IST-FP6-IP-SERENITY, and 2003-S116-00018 PAT-MOSTRO.

References

Anderson, R. J. (1994). Why cryptosystems fail. *Communications of the ACM*, 37(11):32–40.

Association Cambiste Internationale (2005). The Model Code: The International Code of Conduct and Practice for the Financial Markets . http://www.aciforex.com/market/July05_ModelCode.pdf.

Axelrod, R. (1984). *The Evolution of Cooperation*. Basic Books.

Bandara, A. K., Lupu, E., and Russo, A. (2003). Using Event Calculus to Formalise Policy Specification and Analysis. In *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks*, pages 26–39. IEEE Computer Society Press.

Basin, D., Doser, J., and Lodderstedt, T. (2006). Model Driven Security: from UML Models to Access Control Infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15(1):39–91.

Bell, D. E. and LaPadula, L. J. (1976). Secure Computer System: Unified Exposition and MULTICS Interpretation. Technical Report MTR-2997 Rev. 1, The MITRE Corporation, Bedford, MA.

Blomqvist, K. and Ståhle, P. (2000). Building Organizational Trust. In *Proceedings of 16th Annual IMP Conference*.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

Castelfranchi, C. and Falcone, R. (1998). Principles of trust for MAS: Cognitive anatomy, social importance and quantification. In *Proceedings of 3rd International Conference on Multi-Agent Systems*, pages 72–79. IEEE Computer Society Press.

Cheng, B. H., Konrad, S., Campbell, L. A., and Wassermann, R. (2003). Using Security Patterns to Model and Analyze Security Requirements. Technical Report MSU-CSE-03-18, Department of Computer Science, Michigan State University.

Clark, D. D. and Wilson, D. R. (1987). A Comparison of Commercial and Military Computer

Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 184–195. IEEE Computer Society Press.

Dardenne, A., van Lamsweerde, A., and Fickas, S. (1993). Goal-directed Requirements Acquisition. *Science of Computer Programming*, 20:3–50.

Doan, T., Demurjian, S., Ting, T. C., and Ketterl, A. (2004). MAC and UML for secure software design. In *Proceedings of the 2004 ACM workshop on Formal Methods in Security Engineering*, pages 75–85. ACM Press.

Downs, D., Rub, J., Kung, K., and Jordan, C. (1985). Issues in Discretionary Access Control. In *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, pages 208–218. IEEE Computer Society Press.

Giorgini, P., Massacci, F., Mylopoulos, J., and Zannone, N. (2005a). Modeling Security Requirements Through Ownership, Permission and Delegation. In *Proceedings of the 13th IEEE International Requirements Engineering Conference*, pages 167–176. IEEE Computer Society Press.

Giorgini, P., Massacci, F., Mylopoulos, J., and Zannone, N. (2005b). Modelling Social and Individual Trust in Requirements Engineering Methodologies. In *Proceedings of the Third International Conference on Trust Management*, volume 3477 of *Lecture Notes in Computer Science*, pages 161–176. Springer.

Giorgini, P., Massacci, F., Mylopoulos, J., and Zannone, N. (2006). Requirements Engineering for Trust Management: Model, Methodology, and Reasoning. *International Journal of Information Security*. To appear.

Giorgini, P., Massacci, F., and Zannone, N. (2005c). Security and Trust Requirements Engineering. In *Foundations of Security Analysis and Design III - Tutorial Lectures*, volume 3655 of *Lecture Notes in Computer Science*, pages 237–272. Springer.

Johnson, C. W. (2005). V2: Using Violation and Vulnerability Analysis to Understand the Root Causes of Complex Security Incidents. Submitted to ACM Trans on Information and System Security.

Jorion, P. (2000). *Value-at-Risk: The New Benchmark for Managing Financial Risk*. McGraw-Hill.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. (2005). The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*.

Lupu, E. C. and Sloman, M. (1999). Conflicts in Policy-Based Distributed Systems Management. *IEEE Transactions on Software Engineering*, 25(6):852–869.

Massacci, F., Prest, M., and Zannone, N. (2005). Using a Security Requirements Engineering Methodology in Practice: The compliance with the Italian Data Protection Legislation. *Computer Standards & Interfaces*, 27(5):445–455.

McDermott, J. and Fox, C. (1999). Using Abuse Case Models for Security Requirements Analysis. In *Proceedings of 15th Annual Computer Security Applications Conference*, pages 55–66. IEEE Computer Society Press.

McKnight, D. H. and Chervany, N. L. (1996). The meanings of trust. Technical Report 96-04, MIS Research Center.

Moffett, J. D. and Sloman, M. S. (1994). Policy Conflict Analysis in Distributed System Management. *Journal of Organisational Computing*, 4(1):1–22.

Nyanchama, M. and Osborn, S. (1999). The role graph model and conflict of interest. *ACM Transactions on Information and System Security*, 2(1):3–33.

Ponemon, L. (2003). What Keeps Security Professionals Up At Night? URL: http://www.darwinmag.com/read/040103/threats.html.

Priebe, T., Fernández, E. B., Mehlau, J. I., and Pernul, G. (2004). A Pattern System for Access Control. In *Proceedings of the Eighteenth Annual Conference on Data and Applications Security*, pages 235–249. Kluwer.

Promontory Financial Group, Wachtell, Lipton, Rosen, and Katz (2003). Report to the Board and Directors of Allied Irish Bank P.L.C., Allfirst Financial Inc., and Allfirst Bank Concerning Currency Trading Losses.

Ray, I., Li, N., France, R., and Kim, D.-K. (2004). Using UML to visualize role-based access control constraints. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies*, pages 115–124. ACM Press.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2):38–47.

Schaad, A. and Moffett, J. D. (2002). A lightweight approach to specification and analysis of role-based access control extensions. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies*, pages 13–22. ACM Press.

Schumacher, M. (2003). *Security Engineering with Patterns*, volume 2754 of *Lecture Notes in Computer Science*. Springer.

Schumacher, M., Fernandez, E. B., Hybertson, D., Buschmann, F., and Sommerlad, P. (2005). *Security Patterns - Integrating Security and Systems Engineering*. John Wiley & Sons.

Shin, M. E. and Ahn, G.-J. (2000). UML-Based Representation of Role-Based Access Control. In *Proceedings of the 9th IEEE International Workshops on Enabling Technologies*, pages 195–200. IEEE Computer Society Press.

Simon, R. and Zurko, M. E. (1997). Separation of duty in role-based environments. In *Proceedings of the 1997 IEEE Computer Society Security Foundations Workshop*, pages 183–194. IEEE Computer Society Press.

Sindre, G. and Opdahl, A. L. (2005). Eliciting security requirements with misuse cases. *Requirements Engineering Journal*, 10(1):34–44.

The Open Group (2004). Security Design Patterns – Technical Guides.

United States Department of Justice (2002). United States of America v. John M. Rusnak. SMS/SD/USAO #2002R02005. http://www.usdoj.gov/dag/cftf/chargingdocs/allfirst.pdf.

van Lamsweerde, A., Brohez, S., De Landtsheer, R., and Janssens, D. (2003). From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering. In *Proceedings of the 2nd International Workshop on Requirements for High Assurance Systems*, pages 49–56.

van Lamsweerde, A., Darimont, R., and Letier, E. (1998). Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24(11):908–926.

Yu, E. S.-K. (1996). *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto.