Security Obstacle Detection and Response for Secure Information Systems

Dehousse, Stéphane; Faulkner, Stephane; Jureta, Ivan; Mouratidis, Haramblos; Giorgini, Paolo

Publication date:
2008

Document Version
Early version, also known as pre-print

Security Obstacle Detection and Response for Secure Information Systems

S. Dehousse[1], S. Faulkner[1], I. J. Jureta[1], H. Mouratidis[2], P. Giorgini[3]

[1] PReCISE Research Center - University of Namur, Belgium
[2] School of Computing and Technology, University of East London, UK
[3] Department of Information and Communication Technology, University of Trento
stephane.dehousse@fundp.ac.be, stephane.faulkner@fundp.ac.be, iju@info.fundp.ac.be, H.Mouratidis@uel.ac.uk, paolo.giorgini@unitn.it

Abstract. Securing an information system (IS) involves continual detection and response to security obstacles at all stages of the system's lifecycle. During early requirements engineering, one way to facilitate these tasks is to use secure actor and goal models that represent desired IS behaviour, including responses to security obstacles. We propose the Security Obstacle Detection and Response (SODR) method for detecting and responding to security obstacles during the construction of secure actor and goal models in early RE. SODR relies on a justification process, in which arguments about potential security obstacles related to IS assets are made explicit, submitted to justification, leading either to the identification of justified responses to the given security obstacles or the rejection of these obstacles. SODR thereby ensures that the arguments behind the identified security obstacles and chosen responses are explicit, so that they can be constructed, openly questioned, and revised in a structured and rigorous manner. The proposed method is supported by the SODR-Tool.

1 Introduction

Securing an information system (IS) involves taking action to continually achieve and maintain security goals, such as confidentiality, integrity, and availability of various assets of the IS. An obstacle is a dual notion to goal [1]: while a goal captures desired conditions, an obstacle captures undesirable, yet possible ones. Security obstacles (SOs) obstruct the achievement of security goals. Securing an IS therefore involves continual detection and response to all SOs that may allow unintended use of services offered by, and resources made available to the IS. Insufficiently secure IS are prone to compromise by malicious users or third parties, leading usually to uncontrolled distribution of sensitive information, waste of already limited resources, and overall damaging consequences. It is evident that insufficiently secure IS can only be considered as low quality systems, therefore unacceptable in the present context of incresing reliance on IS. It is widely recognized that security must be accounted for during the very first, *early* stages of IS development, when early IS requirements are elicited, structured, and analyzed, then over all subsequent development stages, deployment, entire IS runtime, until the system becomes obsolete and is replaced. It is also accepted that, the longer an SO remains undetected during the IS lifecycle, the higher the cost of

its subsequent detection and resolution. It has, however been recently observed [2] that efforts in security modeling and enforcement of security policies (e.g., [3]) have been largely independent of work invested at the early stages of IS development, when requirements are being engineered for the future IS. Nevertheless, current research [4] indicates that modern IS require a unified treatment of security and other requirements and attention to their relationships.

Problem. This paper focuses exclusively on security assurance during *early* requirements engineering (RE). It is necessary at that earliest stage to secure the IS by detecting SOs and finding responses thereto, if we are to reduce the overall cost of the IS throughout its lifecycle. Dealing with security considerations at early RE is not only critical, but also particularly difficult: qualitative and informal information abounds, while some quantitative information may be available (e.g., failure statistics from similar systems), yet all of that information must be accounted for and systematically analyzed in order to appropriately detect and resolve SOs. Several security-oriented approaches to early RE have been proposed in the last couple of years (See, §5) — they all feature modeling primitives, such as goals, dependencies, tasks, and actors, augmented with primitives for the representation of security constraints, security goals to be achieved and security tasks to be executed in order to satisfy security constraints. However, they fail on the following accounts:

- There is no apparent systematic way to use both qualitative and quantitative information expressed informally or formally when drawing conclusions on how and what SOs to address.
- Applying the available approaches leads to the definition of a "secure (actor and goal) model". From the perspective of the IS stakeholders (i.e., owners, users, etc.), there is nothing in the secure actor and goal model to assure them that a rigorous, structured approach has been applied to identify SOs and then define appropriate responses thereto. In absence of any such justification, stakeholders cannot know whether to perceive as appropriate the set of identified SOs and responses. Given the recurrent and widely publicised security problems (e.g., [5]), the IS engineers' reputation alone is no longer a sufficient guarantee.
- For systems that have a long lifespan (e.g., governing the operation of nuclear power plants or space shuttles), secure actor and goal models will be read and revised by various parties at different times. If it remains unknown *why* some response is defined for some SO, then it also remains unknown whether a response that worked until some point remains relevant after new events occur. If we have explicit arguments in favor of a response, we can confront information about new events with the available arguments to decide whether a revision of the response is needed. There are no explicit arguments in secure actor and goal models, and available approaches do not suggest that such arguments be recorded.
- The ideas, arguments, and assumptions underlying the identification of some SO and/or response remain implicit and/or are lost over time. Alternative ideas and confronting views that could lead to finding additional, possibly

more critical SOs are lost as well. Both can lead to a poor understanding of the SOs in the given IS and potential responses thereto.

Addressing these problems during early RE requires us to externalize and record arguments that led the modeling participants to identify and maintain some SOs and disregard others, and define responses in some particular way. That is, by recording arguments, we can confront them and evaluate decisions as either justified or not. In doing so, we must accommodate the fact that early RE involves mostly the treatment of informal and qualitative information, for which structured and rigorous approaches are missing.

Contributions. In response, we introduce the *Security Obstacle Detection and Response* (SODR) method to guide SO detection and definition of responses during the construction of secure actor and goal models in early RE. Drawing on AI argumentation models [6], SODR integrates a justification process, in which arguments for potential SOs of IS assets are made explicit, submitted to justification, leading either to the identification of justified responses to the given SOs or the rejection of SOs. SODR thereby ensures that the arguments behind identified SOs and chosen resolutions are explicit, so that they can be constructed, openly questioned, and revised in a structured and systematic manner. The use of arguments and justification allows us to integrate all relevant early information, be it expressed informally in natural language or in a mathematical notation, and regardless of it being quantitative or qualitative. SODR is not designed to replace available secure actor and goal modeling approaches, but to complement them where they are weak - it is then up to the IS engineer to choose the early goal-oriented RE modeling approach to security, while using SODR to systematically detect and resolve SOs. In this paper, SODR is applied within the context of the Tropos and Secure Tropos methodologies [7]. We have opted for such approach since Tropos employs a set of concepts that are common in early RE while Secure Tropos complements these concepts with a number of security related ones. A real-life case study on a health and social care IS serves for illustration.

Organization. We start our analysis by considering the relationships between the different actors in our case study environment using Tropos actor models (§2). SODR is then applied, and the steps of the method explained and illustrated to identify SOs and responses thereto (§3). We discuss SODR and present the toolset developed to support the users of SODR (§4). After related efforts are reviewed (§5), we close the paper with conclusions, limitations, and pointers to future work (§6).

2 Pre-SODR Tropos Modeling for the Case Study

During early RE, the IS engineer (i) represents the (organizational) environment in which the IS will be introduced, and (ii) studies the probable impact of the future IS on this environment, that is, identifies the purpose of the IS, its desired behavior, and the system's main interactions and interdependencies with its environment and stakeholders. Goal-oriented security-sensitive models employed at early RE feature modeling primitives, such as goals, dependencies, tasks, and

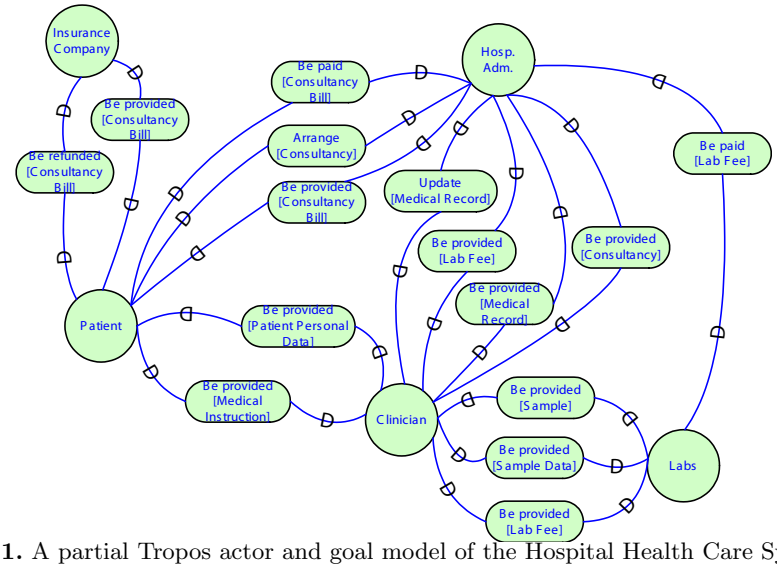

Fig. 1. A partial Tropos actor and goal model of the Hospital Health Care System.

actors, augmented with primitives for the representation of security constraints, security goals to be achieved and security tasks to be executed in order to satisfy security constraints. *Secure Tropos* [7] is an RE methodology that extends the *Tropos* [8] RE methodology with notions needed for the definition of constraints on the desired behavior of the future IS. The aim of these constraints is to secure the future IS by restricting its behavior so that SOs are avoided and/or responses to unavoidable SOs activate when needed.

For the case study, we consider the health and social care domain in which IS facilitate the provision of health and social care services. Integrated health and social care IS are being increasingly considered as a means for providing more effective health care to the elderly. In the UK, the electronic Single Assessment Process (eSAP) system [9] is one such effort. Overall, the IS aims to automate care processes, including, e.g., assessment procedures, collection and management of information about the elderly, the scheduling of appointments, and so on. An important related system is the The Hospital Health Care IS, which aims to help maintain good patient (in this case, an elderly person) health by ensuring that the patient receives good care from the clinician. An actor and goal model for such a system is represented using Tropos in Figure 1. It models actors, such as *Patient* and *Clinician*, and their interdependencies in the achievement of goals. E.g., the Patient expects the Clinician to provide the Medical Instructions appropriate with regards to the Patient's condition. The Clinician cannot provide advice to the Patient if the patient's personal data is missing, so that the Clinician depends on the Patient for providing the personal data. Goals, such as *Be provided [Medical Record]* are represented with rounded rectangles. In the form shown in Figure 1, the model is not concerned with SOs or responses. Therefore, we emply SODR to identify and justify the relevant SOs and responses. With SODR, we are able to identify potential SOs from the initial

Tropos actor and goal models, analyze them and identify justified responses, leading us to construct a Secure Tropos actor and goal models in which SOs and responses are explicit.

3 Security Obstacle Detection and Response

Overview of SODR. SODR proceeds in four steps:

1. The first step of SODR (§3.1) is the identification of assets using the model provided by the goal-oriented RE framework (here, we use Tropos) chosen by the IS engineer. Assets are decomposed and classified. Depending on the classification, the body of knowledge available in security-related standards is used to determine potential SOs that may arise in relation to each asset.
2. In the second step (§3.2), the engineer considers each potential SO, sets it as the root of an argumentation tree, then search for arguments to support the occurrence of that SO. Each favorable argument indicates a reason to believe that the SO will indeed occur. The search for arguments is a step-by-step process, in which one first seeks arguments about how the actions of actors, who can access the asset, might lead the SO to occur. The engineer then look for motives for these actors to indeed act in the aim of realizing the identified SOs. Knowing the potential damaging motivations, circumstances are sought in which these motives may appear.
3. Once the circumstances are known, the engineer identifies at the third step (§3.3) the responses which will adjust the future system's behavior so that the circumstances can be avoided. Each response defines a new argumentation tree, is set as the root of that tree, and the engineer proceeds to justify the relevance of the response with regards to the relevant SO. In other words, arguments are sought that support or counterargue the proposed response. In this way, the engineer can justify the choices of responses.
4. At the last step (§3.4), the engineer uses a goal-oriented security-sensitive RE framework (here, we use Secure Tropos) to build the secure actor and goal models. The secure actor and goal models ensure that the IS responds to the SOs through the justified responses.

Each step is detailed below. First, however, we provide conceptual background needed to understand our models of arguments and the justification process, both extensively used in SODR. Our definitions are formal when possible, to ensure that SODR relies on solid and precise conceptual foundations.

Argumentation and Justification. An *argumentation model* is a static representation of an *argumentation process*, which can be seen as a search for arguments, where an argument consists of a set of rules chained to reach a conclusion. Each rule can be rebutted by another rule based on new information. To formalize such defeasible reasoning, elaborate syntax and semantics have been developed (e.g., [6, 10]) commonly involving a logic to formally represent the argumentation process and reason about argument interactions. *Note that the definitions below provide formal syntax and informal but precise semantics for argumentation trees used in the justification process, and thereby for the graphical representation of argumentation trees used in Figures 3, 4, and 5.*

Definition 1. *An argument is defined recursively as follows:*

1. *Any expression of the form $\langle P, c\rangle$ is an argument, where c is called "conclusion" and is a speech act of any type (i.e., assertive, directive, commissive, expressive, or declarative) and P is a set of assertive propositions suggested to support the conclusion (such a proposition is called "premise").*
2. *If for A and B such that $A = \langle P_A, c_A\rangle$ and $B = \langle P_B, c_B\rangle$, $P_A \subseteq P_B$ then A is a subargument of B.*
3. *An argument cannot have its conclusion as a premise for its conclusion: if $A = \langle P_A, c_A\rangle$ and $c_A \in P_A$ then A is not an argument.*
4. *There can be no inconsistent propositions in P.*
5. *Nothing is an argument unless it obeys the rules above.*

The suggested definition is common in philosophy (see, e.g., [11] for a discussion) and AI (for an overview, see [6]). It bans inconsistent information as support for a conclusion, allows complex arguments, in which a premise can be a conclusion of another argument, and bans cyclical argumentation.

Definition 2. *A **justification** $\langle P, c\rangle$ is an argument that remains undefeated after the justification process.*[4]

Definition 3. *The **justification process** [10] consists of recursively defining and labeling a dialectical tree $\mathcal{T}\langle P, c\rangle$ as follows:*
1. A single node containing the argument $\langle P, c\rangle$ with no defeaters is by itself a dialectical tree for $\langle P, c\rangle$. This node is also the root of the tree.
2. Suppose that $\langle P_1, c_1\rangle, \ldots, \langle P_n, c_n\rangle$ each defeats[5] *$\langle P, c\rangle$. Then the dialectical tree $\mathcal{T}\langle P, c\rangle$ for $\langle P, c\rangle$ is built by placing $\langle P, c\rangle$ at the root of the tree and by making this node the parent node of roots of dialectical trees rooted respectively in $\langle P_1, c_1\rangle, \ldots, \langle P_n, c_n\rangle$.*
3. When the tree has been constructed to a satisfactory extent by recursive application of steps 1 and 2 above, label the leaves of the tree undefeated (U). For any inner node, label it undefeated if and only if every child of that node is a defeated (D) node. An inner node will be a defeated node if and only if it has at least one U node as a child. Do step 4 below after the entire dial. tree is labeled.
4. $\langle P, c\rangle$ is a justification (or, P justifies c) iff the node $\langle P, c\rangle$ is labelled U.

We focus in this paper only on arguments with informal content. We thus use the argument trees along the above definitions but do not exploit here, for lack

[4] Some background [10]: Let A a set of agents (e.g., stakeholders) and the first-order language $\mathcal{L}$ defined as usual. Each agent $a \in A$ is associated to a set of first-order formulae K_a which represent knowledge taken at face value about the universe of discourse, and Δ_a which contains defeasible rules to represent knowledge which can be revised. Let $K \equiv \bigcup_{a\in A} K_a$, and $\Delta \equiv \bigcup_{a\in A} \Delta_a$. "$\mid\!\sim$" is called the *defeasible consequence* and is defined as follows. Define $\Phi = \{\phi_1, \ldots, \phi_n\}$ such that for any $\phi_i \in \Phi$, $\phi_i \in K \cup \Delta^{\downarrow}$. A formula ϕ is a defeasible consequence of Φ (i.e., $\Phi \mid\!\sim \phi$) if and only if there exists a sequence $B_1, \ldots, B_m$ such that $\phi = B_m$, and, for each $B_i \in \{B_1, \ldots, B_m\}$, either B_i is an axiom of $\mathcal{L}$, or B_i is in Φ, or B_i is a direct consequence of the preceding members of the sequence using modus ponens or instantiation of a universally quantified sentence. An argument $\langle P, c\rangle$ is a set of consistent premises P supporting a conclusion c. The language in which the premises and the conclusion are written is enriched with the binary relation $\hookrightarrow$. The relation $\hookrightarrow$ between formulae α and β is understood to express that "reasons to believe in the antecedent α provide reasons to believe in the consequent β". In short, $\alpha \hookrightarrow \beta$ reads "α is reason for β" (see, [10] for details). Formally then, P is an argument for c, denoted $\langle P, c\rangle$, iff: (1) $K \cup P \mid\!\sim c$ (K and P derive c); (2) $K \cup P \nvdash \bot$ (K and P are consistent); and (3) $\nexists P' \subset P, K \cup P' \mid\!\sim c$ (P is minimal for K).

[5] Roughly (for a precise definition, see [10]) the argument $\langle P_1, c_1\rangle$ *defeats at* c an argument $\langle P_2, c_2\rangle$ if the conclusion of a subargument $\langle P, c\rangle$ of $\langle P_2, c_2\rangle$ contradicts $\langle P_1, c_1\rangle$ and $\langle P_1, c_1\rangle$ is more specific (roughly, contains more information) than the subargument of $\langle P_2, c_2\rangle$.

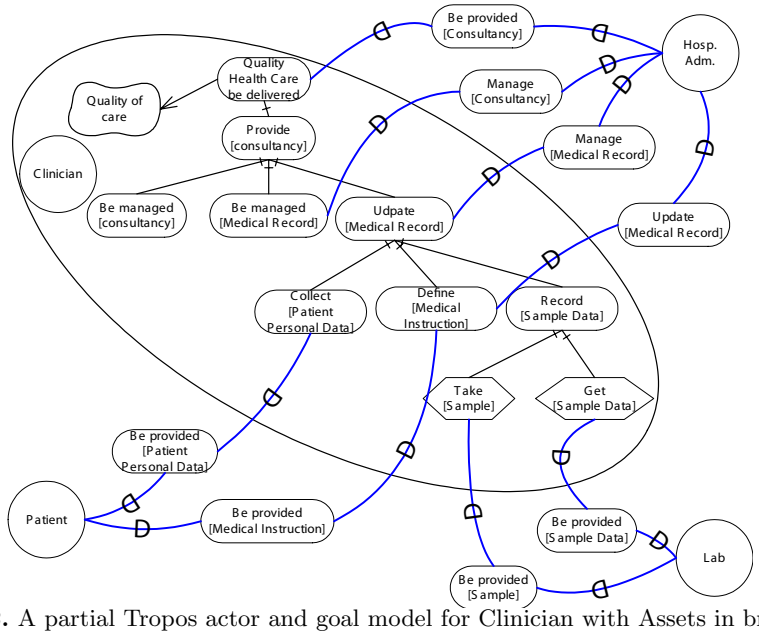

Fig. 2. A partial Tropos actor and goal model for Clinician with Assets in brackets

of space, the possibility for, e.g., automated detection of argument conflict. The informal approach below is especially relevant at the early RE stage, when it often proves too early in practice to proceed to formalization. A separate paper will evaluate the feasibility and cost of formalizing arguments at early RE and address automated reasoning on argument trees.

3.1 Step 1: Analyzing Assets

Is *Patient Personal Data* (PPD) important for the system? How does *Patient Personal Data* relates to *Medical Record*? To be able to properly answer those questions, engineers must learn to identify assets and trace their dependencies in the IS and its environment. An asset is anything of value in the IS [12]. We rely on the available body of knowledge for asset identification and classification. The set of assets comprises information objects stored in or accesed by the system, tangible objects such as computers themselves [13], etc. The British BS7799 standard [14] suggests to differentiate between information assets, documents, software assets, physical assets, personnel assets, image and reputation, and service (utilities). According to the type of asset, the Common Criteria Project (involving, e.g., NIST, NSA, and others) [12] identifies different categories of security goals for any information asset the C.I.A. triad:

- *Confidentiality* is the assurance of data privacy, e.g. ensure that an asset is visible only to actors authorized to see it.
- *Integrity* is assurance that an asset can only be modified in appropriate ways by specific actors. It generally integrates the assurance of the ability to restore to a trustworthy state with minimum lost after damage or incorrect alteration.
- *Availability* is the assurance that an asset is readily accesssible to actors when needed.

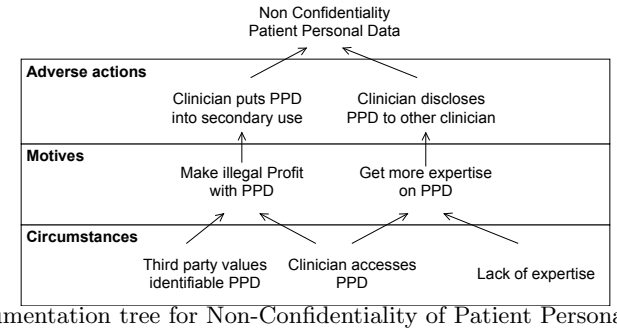

Fig. 3. Argumentation tree for Non-Confidentiality of Patient Personal Data (PPD)

In the Tropos methodology, the actor and goal model gives the analysists a graphical way to elicit assets and their relationships. The model emphasizes assets' related inter-actor relationships through dependencies, and the necessary intra-actor information can be represented to trace assets interdependencies. The model in Figure 1 shows a dependency between the *Clinician* and the *Patient* about *Medical Instruction* and a dependency from the *Hospital Adm* with the *Clinician* about *Medical Record*. *Medical Instruction* and *Medical Record*, being information objects, are assets. The analysis of the model in Figure 2 for the *Clinician* reveals a relationship between the *Medical Instruction* and the *Medical Record*. The exact meaning of the relationships is given by the stakeholders. In this case, an is-part-of relationship exists between the *Medical Instruction* and the *Medical Record*. Same reasoning applied to the other assets reveals that the *Medical Record* has a is-part-of relationships with *Medical Instruction*, *Patient Personal Data* and *Sample Data* assets.

3.2 Step 2: Analyzing Security Obstacles

During this step the engineer builds individual argumentation trees by searching for arguments that support root conclusion elicited in Step 1. This process is facilitated by seeking three main types of arguments. These types are: arguments about adverse actions of actors; arguments about motives leading to the adverse actions; and circumstances in which motives can appear and lead to adverse actions.

Adverse actions. Which actions lead to the violation of the security goal for the asset? Considering any one of the actors as a potential attacker, the engineer identifies adverse actions like committing insurance fraud, hidding malpractice evidence, and putting patient identifiable information into secondary use. An adverse action can also be accidental, e.g. by omittig to update patient data.

Motives. What motives push to actions that create SOs? Actors have reasons for taking adverse actions, and the aim here is to make assumptions about such motives. In the case study, these include, e.g. making illegal profit, or getting a second opinion.

Circumstances. Motives are reinforced in some circumstances (i.e., circumstances can motivate actions). We consider that circumstances encompass actor's capabilities, particular states of the system and/or of the environment. For example, the motive *to make illegal profit with PPD* by *putting PPD into secondary use* can be favored by clinician's capabiliy to access PPD and by third

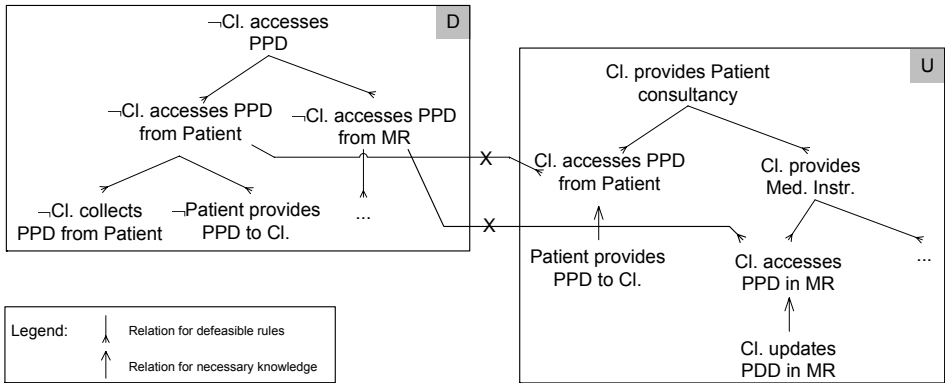

Fig. 4. A dialectical tree for a defeated response

party in the environment that values PPD. Accordingly non-malicous motives like *get second opinion* can be driven by circumstances of a particular situation, e.g. *ambiguous sample data*.

3.3 Step 3: Analyzing Responses

Argumentation trees constructed in Step 2 give reasons to believe that SO will occur in relation to specific security concerns, and for identified assets when the identified circumstances are present. The next step is to define responses to block actions, and/or motives, and/or circumstances so as to avoid SOs. Responses are formulated using the modeling framework of choice: here, we use instances of Secure Tropos concepts to facilitate the modeling at the fourth step below. Namely, a *security constraint* enables the engineer to represent a constraint that limits the behavior of the system and/or stakeholders to that which will not give rise to one or more SOs. We use it therefore to model constraints that will block actions, and/or motives, and/or circumstances. Secure Tropos also introduces the notion of a *secure dependency*, in which a goal or a task is given, whereby the achievement of the goal or the execution of the task ensures that some SO is avoided. With SODR, achieving a goal or executing a task in a secure dependency ensures that insecure actions, and/or motives, and/or circumstances are blocked. Only a justified response is acceptable; that is, a response must remain undefeated after the justification process. For the security constraint *confidentiality of patient personal data*, the engineer define two potential responses as security goals: *Prevent Clinician access PPD* and *Prevent Clinician export PPD*. The argumentation process leads to different arguments that support each responses. For illustration, Figure 4 shows the dialectical tree for *Prevent Clinician access PPD*. The justification process showed that the alternative is unjustified and therefore cannot be accepted. Argument trees for ¬*Clinician accesses PPD* and *Clinician provides consultancy* are shown. The argument $\langle\{\ldots\}, \textit{Clinician provides consultancy}\rangle$ defeats $\langle\{\ldots\}, \textit{Clinician accesses PPD}\rangle$ at *Clinician access PPD from Patient*. Building dialectical trees for response justification can be faciliated if the engineer looks for three types of arguments. These are arguments of early requirements, late

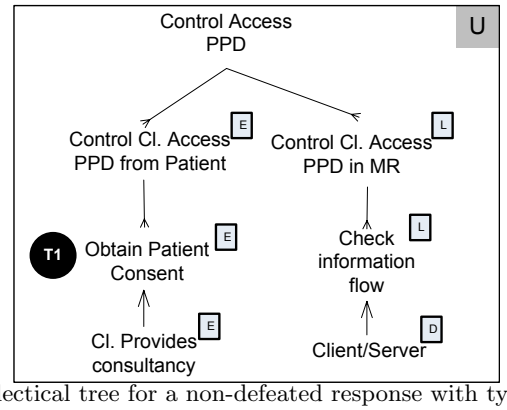

Fig. 5. A dialectical tree for a non-defeated response with typed arguments

requirements and architectural design. Early requirements arguments are security constraints imposed on the actors and their related security dependencies. Late requirements arguments introduce system's security goals and tasks. Finaly, design arguments identify constraints on the architecture of the system with respect to its security requirements. For illustration, Figure 4 shows the dialectical tree for *Control Clinician access to PPD*. The justification process shows that argument *Control Clinician accesses PPD from Patient* and sub-arguments are security constraints imposed to the actors at early requirements. In contrast, argument *Control Clinician accesses PPD from MR* implies new system's security goal and underlying task of *check information flow* to be taken into account at late requirements phase. Finaly, a constraint on design to favor *client/server* architecture is introduced to accomplish information flow checking. We use simple labelling (square labels) of arguments to highlight which is of early or late requirements, or concerns architectural design. The labels shown as circles serve for tracebility between models, and are used at Step 4 of SODR.

3.4 Step 4: Response Modeling

Given the analysis of Step 3, Step 4 amounts to the modeling of the identified responses in the modeling language of choice, which features security primitives. During Step 3, the engineer made decisions on how to protect system's assets from potential SOs. The results of these decisions are at this last step represented in the Secure Tropos actor and goal model, which is a revised and, so to speak "secure" version of the initial Tropos actor and goal model used during Step 1. For illustration, we only show a partial Secure Tropos actor and goal model in Figure 6. It shows how we model the security constraint of patient personal data confidentiality which restricts clinician goal to collect patient personal data. This is derived directly from the content of Figure 5, i.e., from the "Obtain patient consent" argument. Circular labels are used to relate elements from the argumentation tree that gave rise to the various instances of Secure Tropos modeling primitives. The circular labels thus give us simple traces between models used at Steps 3 and 4: the circular label "T1" in Figures 5 and 6 relates the element of the argumentation tree that led the engineer to introduce a particular element in the Secure Tropos actor and goal model. The engineer

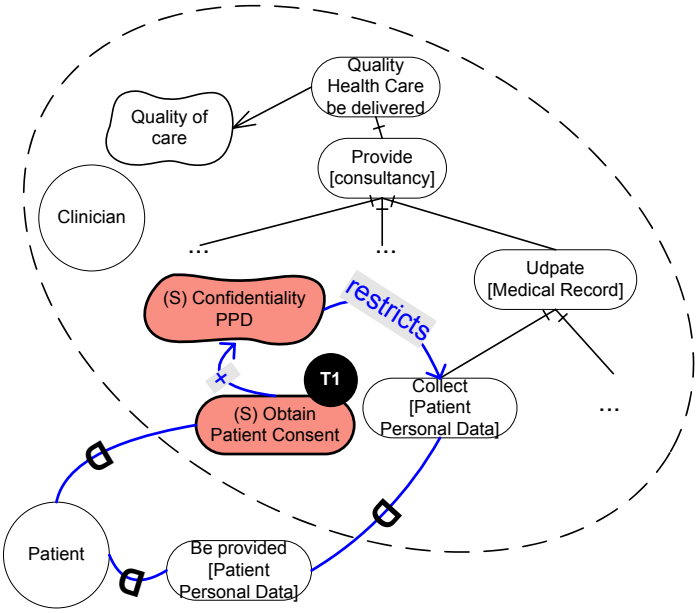

Fig. 6. Revised Goal Diagram for the Clinician

relies on own domain knowledge and knowledge of Secure Tropos to determine how to convert fragments of argumentation trees into fragments of the Secure Tropos model (in our example: to represent the "Obtain patient consent" as a goal in the Secure Tropos model).

4 Evaluation and Tool Support

Our evaluation of the proposed SODR method has raised a number of issues. As already observed in design rationale research (e.g., [15]), recording the rationale of decisions requires additional effort from the engineer and the stakeholders involved. SODR responds by using a simple and intuitive justification process, in which few constraints are placed on how the arguments are written down. The content of arguments, however, should be as precise as possible and checks for rigour in this respect fall on the engineer. Applying argumentation and justification is not trivial, and dialectical reasoning is in general difficult, requiring considerable rigour in thinking. It therefore usually requires some traning for the SODR users. Effort can be limited through automated assistants that suggest which arguments to attack during the justification process.

In response to the practical difficulties we have observed in the use of SODR, we have developed a tool to support the recording of initial Tropos actor and goal models, the argument trees, and the final Secure Tropos actor and goal models. The so-called *SODR-Tool* (screenshots shown in Figure 7) supports the engineer in all steps of SODR. The tool involves the graphical and data layer. The graphical layer provides various modeling primitives used in SODR. The data layer

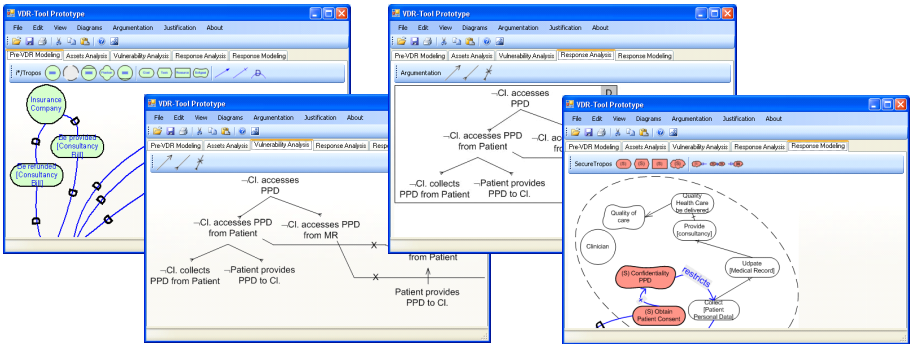

Fig. 7. The SODR-Tool

maintains the collection of data objects separated from graphical layer to enable reuse of common concepts across projects in which SODR is used. The engineer can search through textual information in all models and trees. Elements of the models can be labeled and the same labels applied across several models: we use the labels to indicate (i) that an asset in the Tropos actor and goal model is referenced in an argumentation tree, and (ii) that a fragment of an argumentation tree gives rise to a particular security constraint in the Secure Tropos goal model. That way, we keep traces between models and trees used at various steps of SODR. The argument visualization module supports argumentation at Steps 2 and 3 by procuding "box and arrow" diagrams in which premises and conclusions are formulated as statements. These are represented by nodes that can be joined by lines to display inferences. This module also incorporates a mathematical toolkit that allows the detection of relationships between arguments (i.e., subargument, disagreement, counterargumentation, and defeat, according to definitions from Simari and Loui [10]) in the argument base, provided that the arguments are written in a predicate logic.

5 Related Work

SODR focuses on the recording of the arguments and the justification behind the detection of SOs and the definition of responses thereto. SODR is compatible with and complementary to existing frameworks for security requirements [13, 16, 17], system requirements [8, 7, 18] and risk analysis [19]. Moreover, the concepts and definitions used in SODR follow the terminology proposed by security standars like British BS7799 standard [14], ISO/IEC 2382 [20] and Common Criteria for information technology security evaluation [12].

We now consider efforts comparable to SODR. A number of approaches to securing IS during modeling with the Unified Modeling Language (UML) have been proposed. Jurjens [21] extends UML to enable the analysis of UML diagrams for security issues and responses thereto. Lodderstedt and colleagues [22] provide a modeling language for security to facilitate the specification of access control information and the automatic generation of the necessary in-

frastructure. McDermott and Fox [23] add the so-called abuse case model to complement UML use case modeling. Actors realizing abuse cases are malicious users, whose actions result in security problems. Sindre and Opdahl [24] also deploy a variant of use cases to represent interactions in which security problems occur. De Landtsheer and colleagues [18] extend the KAOS RE framework for late RE with concepts of anti-models and anti-goals. Anti-models are generated cocurrently with actor and goal models to point out goals that malicious users may have, along with measures countering the anti goals. Haley and colleagues' [13] approach derives security goals from assets. They use argumentation for the verification of systems ability to satisfy security requirements. Oladimeji and colleagues [25] represent security issues as negative softgoals of malicious users. Negative contribution is introduced to highlight that a softgoal negatively affects another goal. They suggest a four step process in which they elicit threats, analyze them, and evaluate counter-measures. Liu and colleagues [17] suggest a methodology for dealing with security and privacy requirements within Tropos. The approach consists of performing security analysis of an actor and goal model in which some actors may be interested in attacking the system and in which any dependency may be a source of security issues. Given attackers' intentions and capabilities, it is possible to identify potential issues both within and outside dependencies. Counter-measures are then defined. Mouratidis and colleagues' [26] extend Tropos with modeling primitives relevant for security. At early RE, a security actor and goal model is first built. Security constraints are imposed to actors to render the dependencies secure, and this by restricting actors' individual goals. This approach has also been integrated with UMLSec to ensure that security constraints identified at the early RE level have correspondent notions at late RE and detailed design, when UML is used by the engineer [27]. Security has also been addressed by introducing notions of trust and delegation [28], where trust among actors is studied in order to facilitate the distribution of responsibilities and thereby redefine dependencies. Attack trees have been defined by Schneier in [29] as a formal, methodological way of describing the security of systems, based on varying attacks scenarios. The tree structure is formed by an attack goal (root) and its nodes representing a sequence of attack steps (AND-decomposition) or alternatives ways (OR-decomposition). Such approach is particularly suited for security analysis of existing systems or systems in design stage.

Our work is original in that none of the above approaches considers the recording and analysis of the arguments and the justification process behind the detection of SOs and the definition of responses thereto. Without SODR, arguments behind decisions remain implicit, and lost over time. More importantly, SODR provides a process that is systematic and rigorous, while allowing the treatment of qualitative and quantitative information represented in an informal or formal notation. We have shown throughout the paper the relevance of argumentation and justification for improving the security of IS.

6 Conclusions

Representing and reasoning about security obstacles (SOs) as early in the development process as at the early RE stage, is critical: the cost of responding to SOs missed at early RE only increase as SOs cascade to later development stages. It is also difficult, as early requirements have no predefined format: they can be informally or formally stated, qualitative or quantitative.

We argue that, while there is modeling support for the representation of SOs and responses thereto at early RE, reasoning support can be improved. In response, we propose the *SO Detection and Response* (SODR) method to guide the detection of SO and the definition of responses during the construction of secure models in early RE. SODR relies on a justification process, in which arguments for potential SO of IS assets are made explicit, submitted to justification, leading either to the identification of justified responses to the given SOs or the rejection of SOs. SODR thereby ensures that the arguments behind the identified SOs and chosen responses are explicit, so that they can be constructed, openly questioned, and revised in a structured and rigorous manner.

Future work focuses on the following concerns. Definitions that we use for argument trees and the justification process can be used for basis of automated analysis of argument trees (e.g., automated detection of argument conflicts). This requires, however, that the content of arguments is written in a mathematical logic. We need to evaluate in practice the costs and benefits of doing so, and then, if appropriate work on automated reasoning over arguments by reusing results in applied AI. While useful, the SODR-Tool can be improved to enable preliminary automated checking of argument conflicts by suggesting potentially conflicting arguments by text parsing and subsequent similarity measurement between the textual content. This is expected to facilitate the use of SODR in large projects. A further important concern is the ordering of arguments — namely, we have observed in practice that not all arguments are considered of equal quality and importance by the stakeholders. We are investigating methods for improving the quality of argument content, and we intend to use preference relationships to order arguments and use this information in the justification process. Moreover, we are interested in relating SODR to risk management frameworks.

References

1. van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. IEEE Trans. Software Eng. **26**(10) (2000) 978–1005
2. Devanbu, P.T., Stubblebine, S.G.: Software engineering for security: a roadmap. In: ICSE - Future of SE Track. (2000) 227–239
3. Minsky, N., Ungureanu, V.: Unified support for heterogeneous security policies in distributed systems. In: Proc. Security Conf. (USENIX). (1998)
4. Mouratidis, H., Giorgini, P., eds.: Integrating Security and Software Engineering: Advances and Future Vision. Idea Group, IGI Publishing Group (2006)
5. CERT: Vulnerability remediation statistics. Technical report, Carnegie Mellon University (2007)

6. Chesevar, C.I., Maguitman, A.G., Loui, R.P.: Logical models of argument. ACM Comput. Surv. **32**(4) (2000) 337–383
7. Mouratidis, H., Giorgini, P., Manson, G.A.: When security meets software engineering: a case of modelling secure information systems. Inf. Syst. **30**(8) (2005)
8. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the tropos project. Inf. Syst. **27**(6) (2002) 365–389
9. Mouratidis, H., Philp, I., Manson, G.: Analysis and design of esap: an integrated health and social care information system. In: Proceedings of the International Symposium on Health Information Management Research. (2002)
10. Simari, G.R., Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation. Artif. Intell. **53**(2-3) (1992) 125–157
11. Hitchcock, D.: The concept of argument, and informal logic. In: Philosophy of Logic, Handbook of the Philosophy of Science 5. Elsevier (2006)
12. Criteria, C.: Common criteria for information technology security evaluation version 3.1 rev.1. Technical report, Common Criteria (2006)
13. Haley, C.B., Moffett, J.D., Laney, R., Nuseibeh, B.: A framework for security requirements engineering. In: Proc. SESS '06. (2006)
14. Institute, B.S.: Code of Practice for Information Security Management. BSI (1999)
15. Louridas, P., Loucopoulos, P.: A generic model for reflective design. ACM Trans. Softw. Eng. Methodol. **9**(2) (2000) 199–237
16. T.P., K.: Arguing Safety - A Systematic Approach to Safety Case Management. PhD thesis, University of York (1999)
17. Liu, L., Yu, E., Mylopoulos, J.: Security and privacy requirements analysis within a social setting. In: Proc. RE '03. (2003)
18. van Lamsweerde, A., Brohez, S., Landtsheer, R.D., Janssens, D.: From system goals to intruder anti-goals: Attack generation and resolution for security requirements engineering. In: RHAS'03 Worksh. (2003) 49–56
19. Alberts, C.J., Behrens, S.G., Pethia, R.D., Wilson, W.R.: Operationally critical threat, asset, and vulnerability evaluation (octave) framework, version 1.0. Technical report, Carnegie Mellon University (1999)
20. ISO: ISO/IEC 2382-8: Information technology – Vocabulary – Part 8: Security. International Organization for Standardization (1998)
21. Jurjens, J.: Umlsec: Extending uml for secure systems development. In: Proc. UML '02, London, UK, Springer-Verlag (2002) 412–425
22. Lodderstedt, T., Basin, D., Doser, J.: Secureuml: A uml-based modeling language for model-driven security. In: UML'02. (2002)
23. Mcdermott, J., Fox, C.: Using abuse case models for security requirements analysis. In: Proc. ACSAC '99. (1999)
24. Sindre, G., Opdahl, L.: Eliciting security requirements with misuse cases. Requirements Engineering **10**(1) (2005) 34–44
25. Oladimeji, E., Supakkul, S., Chung, L.: Security threat modeling and analysis: A goal-oriented approach. In: Proc. IASTED. (2006)
26. Mouratidis, H., Giorgini, P., Manson, G.: Modelling secure multiagent systems. In: Proc. AAMAS, New York, NY, USA, ACM Press (2003) 859–866
27. Mouratidis, H., Jurjens, J., Fox, J.: Towards a comprehensive framework for secure systems development. (2006) 48–62
28. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Requirements engineering meets trust management: Model, methodology, and reasoning. In: iTrust. (2004)
29. Schneier, B.: Attack trees. Dr. Dobb's Journal (1999)