Integrating Security Patterns with Security Requirements Analysis Using Contextual Goal Models

Tong Li, Jennifer Horkoff, and John Mylopoulos

University of Trento, Trento, Italy
{tong.li,horkoff,jm}@disi.unitn.it

Abstract. Security patterns capture proven security knowledge to help analysts tackle security problems. Although advanced research in this field has produced an impressive collection of patterns, they are not widely applied in practice. In parallel, Requirements Engineering has been increasing focusing on security-specific issues, arguing for an up-front treatment of security in system design. However, the vast body of security patterns are not integrated with existing proposals for security requirements analysis, making them difficult to apply as part of early system analysis and design. In this paper, we propose to integrate security patterns with our previously introduced goal-oriented security requirements analysis approach. Specifically, we provide a full concept mapping between textual security patterns and contextual goal models, as well as systematic instructions for constructing contextual goal models from security patterns. Moreover, we propose a systematic process for selecting and applying security patterns, illustrated with a realistic smart grid scenario. To facilitate the practical adoption of security patterns, we have created contextual goal models for 20 security patterns documented in the literature, and have implemented a prototype tool to support our proposal.

Keywords: Security Patterns, Security Requirements Analysis, Contextual Goal Model.

1 Introduction

Dealing with security concerns for complex software systems is a laborious and knowledge-intensive process. Security patterns encapsulate reusable security knowledge that can support analysts with little security knowledge. Much work has been done to collect and document such patterns, resulting in several security pattern repositories, such as [1,2,3]. However, such security patterns have not been integrated with existing security requirements analysis techniques, making them difficult to apply as part of early system analysis and design. Analysis using security patterns mainly focuses on "how" to tackle a particular security problem, but does not address "why" a security problem needs to be treated. Effective application of security patterns requires a systematic analysis method, which is currently lacking among existing proposals.

U. Frank et al. (Eds.): PoEM 2014, LNBIP 197, pp. 208–223, 2014.

Requirements Engineering (RE) has been increasingly focusing on security-specific issues, arguing for an upfront treatment of security in software system design. Goal-oriented modeling techniques constitute an effective way to capture and analyze stakeholder intentions. Proposals such as Secure Tropos [4], Secure-i* [5], and STS analysis [6], have been used by multiple authors to analyze security requirements. In this paper, we argue that integrating goal-oriented requirements analysis with security pattern analysis can benefit both types of analysis. Goal models capture the rationale for applying security patterns and facilitate selection among alternatives, while the application of security patterns can efficiently operationalize security requirements into specific security solutions.

Our previous work deals with security requirements for socio-technical systems using a three-layered approach [7]. Our proposed framework does make use of security patterns to assist goal-based security requirements analysis. However, several challenges were revealed during that work, hindering the integration of security patterns and security requirements analysis. Firstly, there is normally more than one security pattern candidate that can potentially treat one security requirement, and analysts have to manually choose the best pattern to apply, a highly non-trivial task. Moreover, the complexity of security pattern selection grows with the number of security requirements. Secondly, creating security patterns in terms of goal models is non-trivial and time-consuming, requiring the analyst to have a full understanding of a security pattern she is about to use. Preliminary steps towards tackling these challenges have been described in a short workshop paper, which proposes to model security patterns as contextual goal models by introducing an initial concept mapping between them [8]. However, the concept mapping was incomplete, as only six security patterns had been analyzed by that time. Also, this work lacks a detailed methodology for selecting and applying security patterns.

In this paper, we significantly improve our previous work [8] in order to seamlessly integrate security patterns with our three-layer security requirements analysis approach [7]. In particular, this paper makes the following contributions:

1. Offers a complete concept mapping between the constituent concepts of security patterns and contextual goal models, as well as a detailed process for constructing contextual goal models from security patterns.
2. Proposes a systematic process for selecting the most appropriate security pattern and applying it to security goal models. The process is illustrated with a realistic smart grid scenario.
3. Sketches a prototype tool, which helps to build contextual goal models and interactively check context. We have built contextual goal models for 20 security patterns adapted from [3] by using this tool.

In the reminder of this paper, we first describe our research baseline on security patterns, contextual goal models, and our previous work in Section 2. In Section 3 we present a smart grid scenario used throughout the paper to illustrate our proposal, while in Section 4 we describe how to transform textual security patterns into contextual goal models. In Section 5, we present a systematic process to select and apply security patterns. We briefly introduce the

prototype tool in Section 6, and then describe related work in Section 7. Finally, conclusions and future work are presented in Section 8.

2 Background

2.1 Security Patterns

Security patterns capture proven security solutions to known security problems. Much work has gone towards identifying security patterns [9,10], while other work focuses on summarizing catalogues of security patterns [11,2,3]. In total, there are more than 100 security patterns.

In this work, we adopt the security patterns presented in [3], which provides detailed specifications of 68 security patterns. In particular, security patterns in [3] are documented in the POSA (Pattern-Oriented Software Architecture) template [12] with predefined sections. Among these sections, there are four essential sections as highlighted in [12]: *Context*, *Problem*, *Force*, and *Solution*. Table. 1 shows the *Intrusion Detection System (IDS)* pattern [3], which contains all four essential sections, as well as the *Consequence* section, which is also analyzed in our approach. In addition to this book, we also took into account several security patterns presented in [2] to cover more security aspects.

2.2 A Goal-Based Contextual Requirements Framework

Stakeholder requirements may vary from context to context. Ali et al. [13] and Lapouchnian et al. [14] argue that requirements should be analyzed in a way that reflects context settings. They propose goal-based frameworks for contextual requirements modeling and analysis, in which they relate goals and contexts.

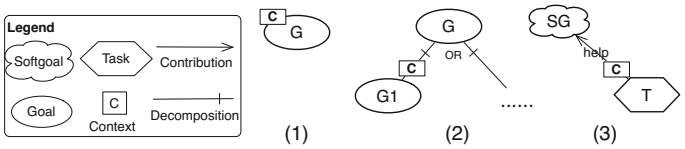

Fig. 1. Goal models with contexts

We use here the proposal by Ali et al. [13], which extends Tropos [15] with context-related concepts. In particular, contexts are treated as labels that can be attached to specific goal model elements, such as what is shown in Fig. 1. Moreover, they define semantics for contexts that are modeled within goal models. For example, in the first case of Fig. 1, goal G represents a requirement if and only if context C applies. In the other two parts of the figure, a link between two goals is part of the goal model only when context C applies. We follow their method to model contexts, but also extend it to suit our purposes.

Table 1. The specification of the Intrusion Detection System pattern [3]

Context: Nodes for local systems that need to communicate with each other using the Internet.
Problem: An attacker may try to infiltrate our system through the Internet. We need to know when an attack is happening and take appropriate response.
Force: • *Incomplete security.* Security measures such as encryption, authentication and so on may not protect all our systems, because they do not cover all possible attacks. • *Non-suspicious users.* Request coming from a non-suspicious address (permitted by a firewall) could still be harmful and should be monitored further. • *Flexibility.* Hard-coding the type of attack can be done easily. But it will be hard and time-consuming to adapt to attack patterns that change constantly.
Solution: Each request to access the network is analyzed to check whether it conforms to the definition of an attack. If we detect an attack, an alert is raised and some countermeasures may be taken.
Consequence: • *Non-suspicious users.* A request coming from a non-suspicious address (permitted by a firewall) is further inspected and analyzed. • *Flexibility.* The detection information can be modified to include new attacks. • There is some overhead in the addition of IDSs to a system.

2.3 A Three-Layer Security Requirements Analysis Framework

Our previous work proposed a three-layer security requirements analysis framework that aims to address security issues at the business layer, software application layer, and physical infrastructure layer [7]. Our framework is designed to support analysis of both security issues within one layer and influences across layers, offering a holistic approach to security analysis.

We use the concept *Security Goal* to represent security requirements. A security goal is specified in a template, *<Importance><Security Property>[<Asset>, <Interval>]*, as shown in Fig. 2. Our approach iteratively carries out security requirements analysis in each of the three layers. In particular, we iteratively refine security goals to "operationalizable" ones, among which we identify critical security goals that need to be treated. Then we operationalize critical security goals into specific security mechanisms, which can fulfill the security requirement being analyzed. Finally, we transfer related influences of selected security mechanisms into the next layer down, and further analyze security requirements in that layer.

We leverage existing security patterns [2,3] to help analysts without security knowledge to operationalize security requirements. In particular, we assign each security pattern a tag, which specifies the security property that a security pattern can potentially tackle. Thus, by matching the security property specified within a critical security goal and the tag of a security pattern, we can identify

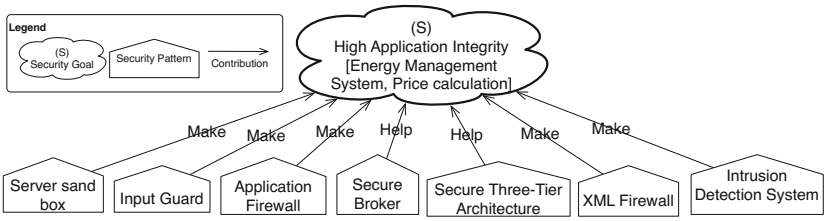

Fig. 2. Security requirements operationalization example

a number of security pattern candidates, among which the analyst must select. For example, as shown in Fig. 2, there are seven security pattern candidates that are identified to tackle the critical security goal *High Application Integrity [Energy Management System, Price calculation]* according to its security property *Application Integrity*. However, the selection among security pattern candidates is a non-trivial task, whose complexity grows with the number of critical security goals. In addition, after choosing a security pattern, the pattern needs to be manually modeled and integrated in the security goal models, which is time-consuming for analysts.

3 An Illustrating Scenario

Our proposed security requirements analysis approach [7] has been applied to a smart grid scenario, which leverages information and communication technologies to enable two-way communications between customers and energy providers. Specifically, the energy provider periodically collects energy consumption data from the customer, based on which they calculate the new price and send back to the customer. Then the customer adjusts his energy usage according to the new price.

We construct a three-layer security requirements goal model for this scenario, consisting of 15 actors, 72 goals, 5 softgoals, 80 tasks, 68 refinement links, 53 operationalization links, and 13 dependency links. Fig. 3 shows part of the requirements goal model within the software application layer. In this paper, our approach will be illustrated using this smart grid scenario.

4 Model Security Patterns as Contextual Goal Models

In this section, we first present a contextual goal modeling language, used to define contextual goal models for security patterns. In addition, we present a detailed process for creating a contextual goal model for a given security pattern. Finally, we summarize some empirical observations derived from modeling 20 security patterns.

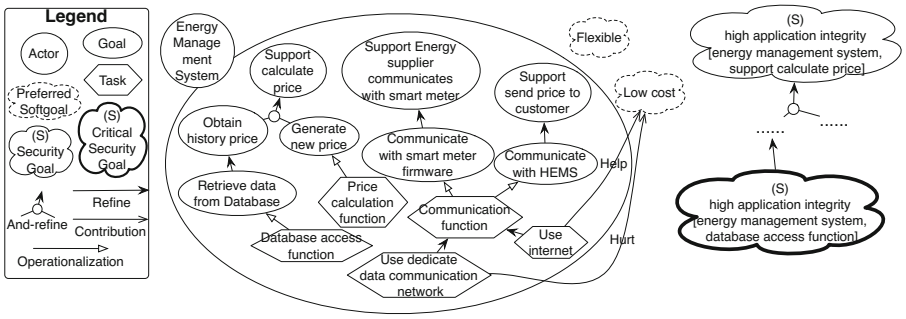

Fig. 3. A part of the security goal model of the smart grid scenario

4.1 A Contextual Goal Modeling Language

We use the goal model concepts of our three-layer requirements analysis [7], including *goal, softgoal, task, domain assumption, refine, and-refine, operationalize, contribution, mandatory, preferred (nice-to-have)*. Specifically, *goals* capture stakeholder desires, while *softgoals* are desires without clear-cut criteria for fulfillment; *tasks* describe behaviors of the system-to-be; and *domain assumptions* are properties of the domain that are assumed to hold. In addition, the *refine* and *and-refine* relations represent the refinement of a requirement into a simpler one; the *operationalize* relation indicates how a goal/softgoal is achieved by the system-to-be; the *contribution* relation captures the influences of *tasks* on *softgoals*; the *mandatory* relation indicates requirements that must be satisfied, while the *preferred* relation indicates "nice-to-have" requirements.

On top of above goal model concepts, we introduce additional concepts to model and analyze contexts within a goal model. In particular, we specify context in terms of *domain properties*. A *domain property* is a fact related to a particular domain, while a *design-time domain property* is a domain property that can be verified at design time by related analysts. For example, "*Computer systems on a local network connected to the Internet*" is a design-time domain property, and analysts can verify this fact during design time according to the designed system infrastructure. For another example, "*The number of users increases significantly*" is not a design-time domain property, as it can only be verified at run-time. Since security pattern analysis is carried out at design-time, we only capture *design-time domain properties* to analyze design-time contexts. A particular context can be arbitrarily complex, consisting of either a single domain property or could be an aggregation of domain properties of any complexity, typically, via *and/or* operators.

As noted in Section 2.2, our goal models are context-dependent. For example, in Fig. 5, the softgoal *Application Security* is required if and only if the context *C1* holds. It is worth noting that the concept *domain assumption* should be distinguished from the concept *design-time domain property*. A *domain assumption* is always assumed to be true during system designs, under which requirements are satisfied, and does not need to be checked. For instance, in Fig. 5, *"other security measures do not cover all possible attacks"* is a domain assumption.

4.2 A Process for Creating Contextual Goal Models from Security Patterns

To build contextual goal models that capture contents of security patterns, we focus on analyzing the five essential predefined sections of security patterns, as illustrated in Table. 1. For each of the five sections of the security patterns, we identify concepts within the contextual goal modeling language to capture the content of the section by considering both of their definitions [3,13]. Our analysis results in a concept mapping, shown in Fig. 4. In the rest of this section, we describe this mapping in detail. Apart from the concept mapping, we further provide detailed guidelines that constitute a systematic process for creating a contextual goal model for a given security pattern. The *Intrusion Detection System* pattern (Table. 1) will be used throughout this section to illustrate the mapping and instructions, with the corresponding contextual goal model shown in Fig. 5.

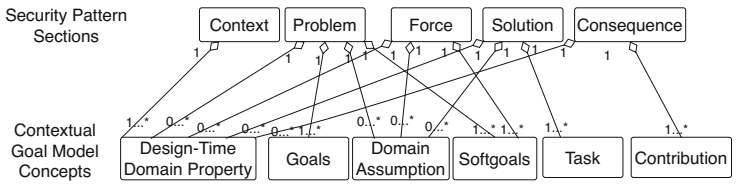

Fig. 4. Concept mappings between contextual goal models and security patterns

Context Section Analysis. This section describes the initial context of the security pattern, in which the security problem occurs and is being solved. We model the context with one or multiple *design-time domain properties*. For example, the context *C1* in Fig. 5 is the initial context, which is represented by one design-time domain property *DTDP1* extracted from the context section. Note that the context *C1*, as the initial context, is attached to the root goal that will be extracted from the *Problem Section*.

Problem Section Analysis. A *problem* is a description of a situation, for which stakeholders do not have a solution. We use one or several *goals* or *softgoals* to capture stakeholder needs concerning such a problem. As the problem is essential to a security pattern, we model the goals/softgoals that capture the problem as *mandatory* requirements, which have to be satisfied by security patterns.

We analyze this section sentence by sentence, each of which usually leads to the inclusion of a goal/softgoal. If there are several goals/softgoals, we need to consider the relations between sentences and determine the refinement structure of these goals/softgoals. It is worth noting that the description of the problem section may also involve domain assumptions, which should be identified and modeled within the refinement structure, such as *"An attacker may infiltrate a system through the internet"* shown in Fig. 5. The root element of a goal model constructed through this process must be mandatory.

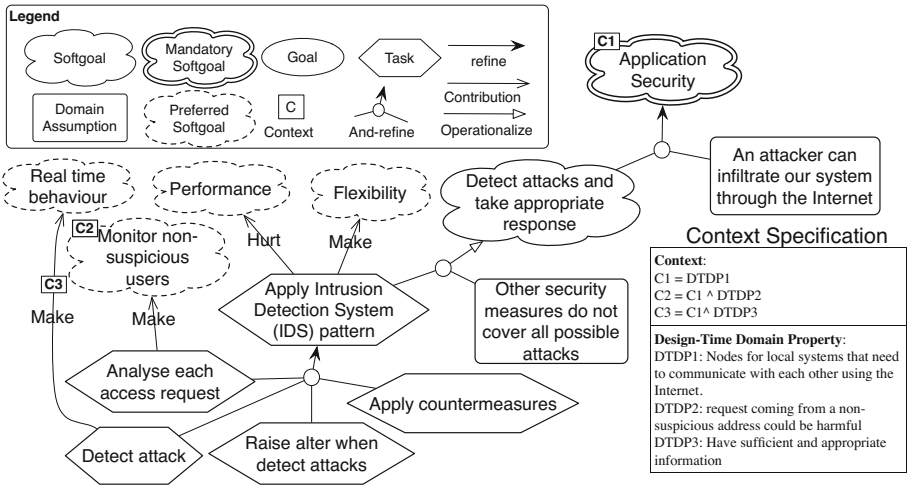

Fig. 5. The contextual goal model of the IDS pattern

To integrate the security patterns with security requirements analysis, we extract not only the specific problems that are solved by the security pattern, but also the high-level security requirements that lead to those detailed problems. If the high-level security requirements are not explicitly specified in this section, we need to do further analysis. In the *IDS* example, the first sentence presents a domain assumption "*Attacker may infiltrate a system through internet*", which also implicitly presents a high-level security requirement that the security of the software application should be protected. The next sentence "*We need to know when an attack is happening*" specifies a detailed problem, which is a refinement of the high-level security requirement. Thus, we obtain a model fragment as shown in the upper-right corner of Fig. 5.

Force Section Analysis. Forces are considerations, often contradictory, which have to be taken into account to determine the applicability of a pattern. These considerations are often related to non-functional requirements (NFRs), such as performance and cost. We model such forces as *preferred softgoals*, where stakeholders want to satisfy as many of such goals as possible. Other forces may belong to domain assumptions, which are always assumed to be true during the security analysis. For instance, the force *Existing security measures cannot cover all attacks*, shown in Fig. 5, is a domain assumption, under which the *IDS* security pattern operationalizes the security goals.

This section is specified in an itemized manner. Each item starts with a key word, from which we can decide whether the force is a preferred softgoal (e.g. *Flexibility*) or a domain assumption (e.g. *Incomplete Security*). It is worth noting that some preferred softgoals are context-dependent, only needing to be considered in particular context. For example, the softgoal "*Monitor non-suspicious users*" only holds under the context "*requests from non-suspicious address could be harmful*". If this context does not hold, the force does not need to consider. Therefore, we identify another context *C2* and add it to this softgoal.

Solution Section Analysis. This section describes actions that are carried out by a security pattern. We model them as *tasks*, which specify how the "system-to-be" implements a security pattern. Similar to the analysis in the *Problem* section, the relations between tasks should also be identified and modeled in an appropriate structure. As shown in Fig. 5, we identify four sub-tasks, which are siblings, for applying the IDS pattern. Note that, the granularity of solutions varies from pattern to pattern. If the information provided in this section is too general, we can optionally extract additional information from other non-essential sections, such as the *Structure*, *Dynamic*, and *Implementation* sections.

Consequence Section Analysis. This section describes the consequences of a security pattern, which indicates both benefits and liabilities of the pattern. We capture these influences using *contribution* links. This section is also documented in an itemized way, and each item should correspond to one force, documented in the *Force* section. However, the correspondence between the *Force* section and the *Consequence* section may not be strict. The *Consequence* section may introduce NFRs in addition to those described in the *Force* section. These NFRs should also be taken into account, via inclusion as preferred softgoals, when choosing a security pattern. For example, the consequence description "*There is some overhead in the addition of IDSs to a system*" (Table. 1) indicates the IDS pattern *hurt*s the performance of a system. The NFR (performance), which is not initially specified in the *Force* section, should be added into our model. In other cases, the preferred softgoals from the *Force* section may not be mentioned in the *Consequence* section. Thus, we need to infer the pattern's influences on those softgoals based on our understanding of the pattern, or search for related knowledge from other reliable knowledge bases.

Some influences on preferred softgoals are also context-dependent. As shown in Fig. 5, the task "*Detect attack*" can only *make* the softgoal "*Real time behaviour*" under the context that there are sufficient and appropriate information about attacks. It is worth noting that the influences of a security pattern may also depend on its detailed implementations. For example, as described in the *Authenticator* pattern, the consequence "*The overhead depends on the protocol used*" cannot be directly modeled. We need to first model two alternative tasks "*Apply a simple protocol*" and "*Apply a complex protocol*", which refines the task "*Apply an appropriate protocol*" and then investigate their influences respectively.

4.3 Pattern Modeling: Empirical Observations

Thus far we have constructed contextual goal models for 20 security patterns[1] described in [3]. During this exercise, we observed several issues, which may affect the quality of resulting models.

1. The specifications of some security patterns are incomplete, such as missing a section.

[1] The full list of models can be found at `http://goo.gl/u539CV`

2. Not all security patterns are specified in a consistent way. For example, some patterns are specified in a threat-oriented manner, while others are in a function-oriented manner.
3. The granularity of descriptions may vary greatly among patterns. For instance, the solution section of some security pattern only describes general idea of the pattern in one sentence, while some other pattern uses several paragraphs to explain related security mechanisms.

These observations disclose that processing and modeling textual security patterns are time-consuming, and additional knowledge related to security patterns is usually required during this process. This fact further explains why security patterns are not widely applied. In the meanwhile, it justifies the value of our work, i.e. constructing *reusable* contextual goal models for 20 security patterns. In addition, the above observations also expose the shortcomings in existing security pattern specifications, which should be tackled by the security pattern community.

5 Integrating Security Patterns with Security Requirements

Once security patterns have been modeled, we follow a systematic process to select and apply them to operationalize critical security goals that are derived from security requirements analysis [7].

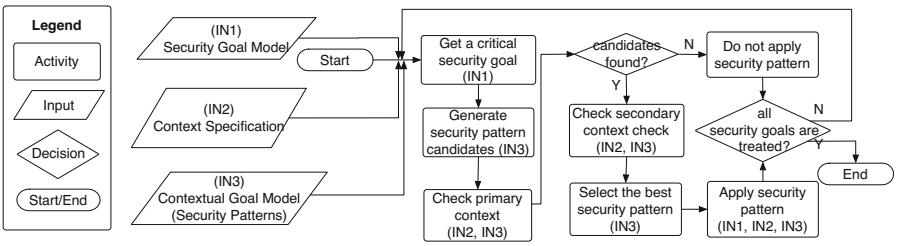

Fig. 6. Security pattern analysis process

As shown in Fig. 6, the process requires three types of information as input: 1) The *security goal model*, which captures requirements and security requirements regarding the domain. In particular, we use our three-layer security goal model (shown in Fig. 3) for this purpose. 2) The *context specification* describes the environments of the domain, which is composed of a list of design-time domain properties. This specification does not need to be complete at the beginning of the analysis, as it can be incrementally enriched during the application of security patterns. 3) A number of *security patterns*, which have been modeled in terms of contextual goal models. Note that, in Fig. 6, each input is assigned a tag, such as

IN1. If the input is required by one activity, the tag of that input will be specified at the end of the description of that activity. The overall analysis process selects the best security pattern for each critical security goal, and applies the selected pattern to the security goal model, as well as updating the context specification. In the rest of this section, we describe each step of the process in detail, and finally illustrate the entire process with the smart grid scenario, described in Sec. 3. In particular, our illustration focuses on the software application layer of the scenario.

5.1 Generating Security Pattern Candidates

The security goal model contains a number of critical security goals, which are analyzed one by one in our analysis process. To apply appropriate security patterns to treat each critical security goal, we first identify security pattern candidates, which can potentially tackle the security goal. Particularly, we match the security property of the security goal with the root goal of the contextual goal model of each security pattern (e.g. *Application Security* in Fig. 5) to determine whether a security pattern can be a candidate solution to the critical security goal. The results of this match will reveal an initial set of security pattern candidates. It is worth noting that the match process takes the hierarchy of security property into account, such as described in [16,2]. For example, the security property *Application Integrity* is a specialization of *Application Security*. Thus, a security pattern that can tackle *Application Security* can also be applied to tackle the *Application Integrity* problem, such as the *IDS* pattern.

5.2 Security Pattern Selection

Once we have the initial set of the security pattern candidates, which typically contains more than one pattern, we carry out context-based selection to choose the most appropriate security pattern. To this end, we need to check each context to determine whether it holds within a particular domain.

As contexts attached to different elements have different effects when they do not hold, we define the context that is attached to root goal of a contextual goal model as a *primary context*, while define the context that is attached to preferred softgoal or contribution links as a *secondary context*. The primary context is essential to the applicability of a security pattern. Such as is shown in the IDS pattern (Fig. 5), here if the primary context *C1* does not hold, the root goal will be deactivated, and the task "*apply IDS pattern*" will be deactivated accordingly, i.e., the pattern becomes inapplicable. In contrast to the primary context, the secondary context mainly affects the quality of the security pattern in terms of its contributions to the preferred softgoals. For instance, in the IDS pattern example, if the context *C2* does not hold, its corresponding preferred softgoal will be deactivated, as well as the contribution links connected to the softgoal.

Having the two types of contexts, we propose two steps for selecting security patterns. As shown in Fig. 6, we first check the primary context of the security

pattern candidates to filter inapplicable security patterns. After that, if there is more than one applicable security pattern left, we check the secondary contexts to determine the quality of each security pattern and do further selection. In particular, we quantify contribution links *{make, help, hurt, break}* as *{2, 1, -1, -2}* respectively to evaluate the effect a pattern has on the satisfaction of preferred softgoals, aiding in selection. Note that other more complicated goal satisfaction analysis techniques can also be used for this selection, such as those compared and evaluated in [17]

As manual checking whether a context applies is a non-trivial task, especially for complex and large models, we propose an interactive process that semi-automates this task. We first formalize check rules for each context in Datalog and automatically check them against the context specification by using our tool. For example, the context *C1* (in Fig. 5) can be formalized as below:

R1: $hold(c1) : -Node(N1), Node(N2), communicate(N1, N2, internet)$

R2: $not\_hold(c1) : -Node(N1), Node(N2), dis\_communicate(N1, N2, internet)$

R3: $undecidable(c1) : -not\ hold(c1), not\ not\_hold(c1)$

If neither hold/not-hold can be inferred for a context, i.e. it is undecidable, the system turns to the user and infers the state of a context on the basis of user answers to a list of yes/no system questions.

5.3 Security Pattern Application

Thanks to the reusable goal model we have constructed for security patterns, analysts do not need to manually construct the goal model of a security pattern each time they want to apply it. Thus, after selecting the best security pattern, the analyst can directly insert the goal model of the security pattern to the security requirements goal model, as illustrated in Fig. 7. Note that the red cross indicates the context *C3* does not hold, and the corresponding contribution link is deactivated. To correctly integrate the goal model of a security pattern into the security goal model, firstly, the analyst needs to merge the softgoals newly introduced by the security pattern with the original softgoals by following the techniques proposed by Niu and Easterbrook [18]. For example, in Fig. 7, the new softgoal *Flexibility* has been merged with the original softgoal *Flexible*. Secondly, the analyst should do a pairwise comparison of all the old elements with all the new elements to find what new contributions should be present. As shown in Fig. 7, two new contributions links are identified with regard to the new softgoal *performance*.

5.4 Case Study Statistics

We apply our approach to the application layer of the three-layer security goal model that is built for the smart grid scenario (Sec. 3) to further exemplify our approach. In this security goal model, we have identified five critical security goals, which need to be treated by specific security patterns. The security properties of these critical security goals, which are essential for initially generating security pattern candidates, are listed in Table. 2.

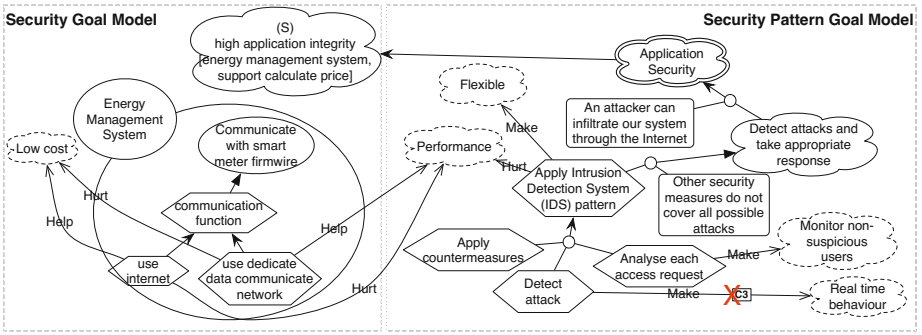

Fig. 7. Applying IDS pattern to the goal model of the smart grid scenario

Table 2. Statistics of applying 20 security patterns to the smart grid scenario

Security Goal	Security Property	All	Candidates	Applicable	Applied
SG1	Application Integrity	20	7	5	1
SG2	Application Integrity	20	7	0	0
SG3	Application Integrity	20	7	2	1
SG4	Data Confidentiality	20	6	4	1
SG5	Application Availability	20	8	0	0

In total we have 20 security patterns that have been modeled as contextual goal models. In the first analysis step, we identify 6-8 security pattern candidates for each of these security patterns. Then, we apply the two-step context analysis to select the most appropriate security pattern, the results of which are shown in Table. 2. Note that some security goals may have no suitable security patterns to apply. In such a case, the security goals will be transferred to the next layer of the three-layer model (physical layer), and are further analyzed there. Finally, we apply each selected security pattern by integrating its goal model into the security goal model.

6 Tool Support

We extend our prototype tool MUSER [19], which has been designed to support the three-layer security requirements analysis approach, with a number of features to implement the approach presented in this paper[2]. The enhanced tool allows us to:

- Graphically model contextual goal models for security patterns;
- Automatically check the contexts of security patterns against the domain-specific context, and ask for manual check if necessary;

[2] Details of the tool can be found at: `http://disi.unitn.it/ li/MUSER/`

– Apply selected security pattern by inserting its corresponding contextual goal models into the security goal models.

7 Related Work

There are several approaches that model security patterns as goal models. Mouratidis et al. extend their security analysis approach Secure Tropos by integrating four security patterns [20]. Yu et al. [21] propose to formally specify role-based access control as a security pattern in terms of i* models, and implement a tool to automatically detect contexts and apply security patterns. However, these approaches do not address the pattern selection issues. In addition, only a limited number of security patterns are presented in their work, and no details are provided on how to model security patterns as goal models.

Araujo and Weiss [22] apply the Non-Functional Requirement (NFR) framework as a complementary representation for security patterns, which helps to analyze the tradeoffs between forces. In particular, they define the *force hierarchy* to represent interactions between forces, and model such hierarchy for 14 security patterns. We can extend their results to build the contextual goal models for those security patterns. Asnar et al. [9] propose a method to design organizational patterns from SI* models, which deals with system security and dependability. Their approach proceeds in the opposite way of ours, they aim to extract security patterns from goal models, while we aim to apply security patterns into goal model analysis.

Security patterns have been applied to various models. Shiroma et al. [23] focus on applying security patterns to UML diagrams, and they define transformation rules to automate this application. Sánchez-Cid and Maña [24] provide a language, which describes security solutions to assist software engineers to implement security patterns into software applications.

Several methods have been discussed for classifying security patterns [1], which are essential for navigating and selecting security patterns, such as security properties, logic tiers, security concepts, system viewpoints and so on. Our approach use security properties for pattern classification. We specify the interrelationships among security patterns by using the refinement and contribution relations, with which we can identify relevant patterns that need to apply.

Other work has also been done on systematically analyzing textual patterns. Gross and Yu [25] specify a systematic way to represent, analyze and apply design patterns by using NFR framework. They illustrate their method and experiences regarding to processing textual design pattern in detail, some of which are similar to our systematic guidelines. However, their approach does not analyze the context of a pattern and provide no support for pattern selections. Supaporn et al. [26] focus on generating security grammars, which are specified in extended-BNF formats, by analyzing descriptions of security patterns. Specifically, they build grammar trees to represent the semantics of security patterns, which could help us to better understand and process the textual security patterns as part of our framework.

8 Conclusions and Future Work

In this paper, we propose to integrate security patterns with our previous security requirements analysis technique by modeling security patterns as contextual goal models. Our approach contributes to both the operationalization of security requirements and the adoption of security patterns. In particular, we define concept mappings between security patterns and contextual goal models, and provide a detailed process for modeling security patterns as contextual goal models. Moreover, we propose a systematic process to select and apply security patterns, and illustrate the process with a realistic smart grid scenario. We have implemented a prototype tool, which supports the application of our approach.

Thus far, we have built contextual goal models for 20 security patterns, and we plan to model more patterns in the future. Our current approach is built on our previous work, and we intend to further generalize it to accommodate other goal-based security requirements analysis techniques. Furthermore, when applying a security pattern model, the newly introduced model may influence different parts of the existing model in various ways and make the application process very hard. Thus, we aim to propose new techniques that facilitate this process. Finally, we plan to carry out further, large-scale empirical evaluations of our approach in order to evaluate its validity and usability.

Acknowledgements. This work was supported in part by ERC advanced grant 267856, titled "Lucretius: Foundations for Software Evolution".

References

1. Hafiz, M., Adamczyk, P., Johnson, R.E.: Organizing security patterns. IEEE Software 24(4), 52–60 (2007)
2. Scandariato, R., Yskout, K., Heyman, T., Joosen, W.: Architecting software with security patterns. Technical report, KU Leuven (2008)
3. Fernandez-Buglioni, E.: Security patterns in practice: designing secure architectures using software patterns. John Wiley & Sons (2013)
4. Mouratidis, H., Giorgini, P.: Secure tropos: a security-oriented extension of the tropos methodology. International Journal of Software Engineering and Knowledge Engineering 17(02), 285–309 (2007)
5. Liu, L., Yu, E.S.K., Mylopoulos, J.: Secure-i*: Engineering secure software systems through social analysis. Int. J. Software and Informatics 3(1), 89–120 (2009)
6. Paja, E., Dalpiaz, F., Giorgini, P.: Managing security requirements conflicts in socio-technical systems. In: Conceptual Modeling, pp. 270–283. Springer (2013)
7. Li, T., Horkoff, J.: Dealing with security requirements for socio-technical systems: A holistic approach. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 285–300. Springer, Heidelberg (2014)
8. Li, T., Mylopoulos, J.: Modeling and applying security patterns using contextual goal models. In: The 7th International i* Workshop, iStar14 (2014)
9. Asnar, Y., Massacci, F., Saidane, A., Riccucci, C., Felici, M., Tedeschi, A., El-Khoury, P., Li, K., Séguran, M., Zannone, N.: Organizational patterns for security and dependability: From design to application. Int. J. Secur. Softw. Eng. 2(3), 1–22 (2011)

10. Fernandez, E.B., Fonoage, M., VanHilst, M., Marta, M.: The secure three-tier architecture pattern. In: CISIS, pp. 555–560 (2008)
11. Schumacher, M., Fernandez-Buglioni, E., Hybertson, D.: Security patterns: Integrating security and systems engineering (2006)
12. Buschmann, F., Henney, K., Schimdt, D.: Pattern-oriented Software Architecture: On Patterns and Pattern Language, vol. 5. John Wiley & Sons (2007)
13. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering 15(4), 439–458 (2010)
14. Lapouchnian, A., Mylopoulos, J.: Modeling domain variability in requirements engineering with contexts. In: Conceptual Modeling-ER 2009, pp. 115–130 (2009)
15. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
16. Firesmith, D.: Specifying reusable security requirements. Journal of Object Technology 3(1), 61–75 (2004)
17. Horkoff, J., Yu, E.: Comparison and evaluation of goal-oriented satisfaction analysis techniques. Requirements Engineering 18(3), 199–222 (2013)
18. Niu, N., Easterbrook, S.: So, you think you know others' goals? a repertory grid study. IEEE Software 24(2), 53–61 (2007)
19. Li, T., Horkoff, J., Mylopoulos, J.: A prototype tool for modeling and analyzing security requirements from a holistic viewpoint. In: The CAiSE 2014 Forum at the 26th International Conference on Advanced Information Systems Engineering (2014)
20. Mouratidis, H., Weiss, M., Giorgini, P.: Modeling secure systems using an agent-oriented approach and security patterns. International Journal of Software Engineering and Knowledge Engineering 16(3), 471 (2006)
21. Yu, Y., Kaiya, H., Washizaki, H., Xiong, Y., Hu, Z., Yoshioka, N.: Enforcing a security pattern in stakeholder goal models. In: Proceedings of the 4th ACM Workshop on Quality of Protection, pp. 9–14 (2008)
22. Araujo, I., Weiss, M.: Linking Patterns and non-functional requirements. In: Proceedings of the Ninth Conference on Pattern Language of Programs (PLOP 2002), September 8-12 (2002)
23. Shiroma, Y., Washizaki, H., Fukazawa, Y., Kubo, A., Yoshioka, N.: Model-driven security patterns application based on dependences among patterns. In: International Conference on Availability, Reliability, and Security 2010, pp. 555–559 (February 2010)
24. Sanchez-Cid, F., Mana, A.: Serenity pattern-based software development life-cycle. In: 19th International Workshop on Database and Expert Systems Application, pp. 305–309 (September 2008)
25. Gross, D., Yu, E.: From non-functional requirements to design through patterns. Requirements Engineering 6(1), 18–36 (2001)
26. Supaporn, K., Prompoon, N., Rojkangsadan, T.: An approach: Constructing the grammar from security pattern. In: Proc. 4th International Joint Conference on Computer Science and Software Engineering (2007)