Modeling and Applying Security Patterns Using Contextual Goal Models

Tong Li, John Mylopoulos

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

Abstract. Security patterns have been proposed to help analysts with little security knowledge to tackle repetitive security design tasks. Although advanced research in this field has produced an impressive collection of patterns, they are not well integrated with security requirements analysis and not easy to apply. Goal-oriented modeling languages have been proposed as an effective way to capture requirements, including security requirements, for socio-technical systems. In this paper, we argue that modeling and analyzing security patterns in contextual goal models can facilitate their applications and magnify their influences in system security design. Particularly, we present a mapping between security patterns and contextual goal models, and provide practical guidelines for transforming textual security patterns into the goal models. In addition, we propose a systematic process for applying security patterns, and discuss how it can be combined with existing security requirements analysis approaches.

1 Introduction

As the complexity of systems increases, system security design is becoming increasingly laborious and requires specialized knowledge about security. Security patterns encapsulate reusable security knowledge, and as such assist analysts in designing secure systems with proven security solutions. Much work has been done to collect and document security patterns, resulting in several security pattern repositories, such as [3]. Security patterns constitute a particular class of design patterns. Although design patterns have been successfully applied in many industrial projects, security patterns have not been widely used [3]. One of the reasons for this is the lack of integration with security requirements analysis techniques, i.e. the analysis regarding to security patterns mainly focuses on solutions rather than problems. Another reason is the lack of a methodology for systematically applying security patterns.

Goal-oriented modeling languages have been proposed as an effective way to capture and analyze system requirements. Many goal-oriented security requirements approaches have been proposed, some of which already applied security patterns in their analysis, such as Secure Tropos [5]. However, this work does not consider contexts and tradeoffs between forces during the selection of security patterns. We argue that integrating goal model analysis and security pattern analysis by modeling security patterns in contextual goal models can benefit both goal-oriented security analysis and security pattern analysis.

Goal model analysis captures the rationale for applying security patterns and facilitate selection among alternative security patterns, while applying security patterns efficiently operationalizes security requirements into detailed security specifications that consists of proven security solutions.

In this paper, we present a goal-oriented modeling language, which extends our previous work [4] with context-related concepts. Based on this language, we transform practical security patterns that are documented in text [3] into contextual goal models. Additionally, we provide a number of guidelines, which facilitate this transformation. Finally, we propose a systematic process to analyze and apply security patterns.

2 Baseline

Our previous work proposed a three-layer security requirements analysis framework, which aims to address security issues in business layer, software application layer, and physical infrastructure layer respectively [4]. In each layer, we not only analyze security requirements but also identify security mechanisms, which can appropriately treat the critical security requirements. We believe that the security mechanisms we apply in one layer shed light on the security requirements in the next layer down, i.e. the upper-layer security requirements indirectly influence the security requirements in the lower-layers. For example, if an analyst designs an auditing activity to treat the security requirements regarding to a data asset in the business layer, the security of this activity itself is also important. Thus, corresponding security requirements should also be applied to the application that implement the auditing activity. Our three-layer security analysis takes the functional requirements models of each of the three layers as input, and iteratively analyzes security issues in each individual layer via four steps: security requirements refinement, simplification, operationalization, and cross-layer analysis. This framework presents a three-layer requirements goal modeling language, which involves concepts *actor, goal, softgoal, task, domain assumption, inference, contribution, dependency and priority*. Those concepts are imported from *i** and *Techne*, and are used in the same way as defined in existing work.

Within the three-layer framework, we leverage existing security patterns to operationalize critical security goals into appropriate security mechanisms. Table. 1 shows a part of the security pattern *Authenticator* [3], which is documented in text. Specifically, our approach matches identified critical security goals with the problem described in a security pattern, and then provides a list of successfully matched security patterns, amongst which the analyst must select. However, this approach easily results in a number of alternative security mechanisms for each critical security goal, and the selection among them is a non-trivial task. In addition, after choosing a security pattern, the current approach requires analysts to manually model and analyze the security patterns according to [3], which is time-consuming and may have different when performed by different people. To tackle these problems, we improve the security pattern analysis in the following aspects. 1) We suggest more suitable security pattens to analysts by taking into account the context of security patterns, in which the security problems happen and the security solutions apply. As shown in Table. 1, if the *Context* does not hold, the security pattern is not applicable. To model context in goal models, we base our approach on existing work [1]. 2) We develop unified and reusable goal model snippets for each security pattern with detailed solutions.

Table 1: Security pattern example — Authenticator [3]

Context: Computer systems contain resources that include valuable information about business plans, user medical records and so on.
Problem: How can we prevent imposters from accessing our system? A malicious attacker could try to impersonate a legitimate user to gain access to their resources. How do we verify that a user intending to access the system is legitimate?
Force: *Flexibility.* A variety of users require access to the system. We need to be able to handle all this variety appropriately. *Performance.* If authentication needs to be performed frequently, performance may become an issue.
Solution: Use a single point of access to receive the interactions of a subject with the system Apply a protocol to verify the identity of the subject. The protocol used may be simple or complex, depending on the needs of the application.

3 Modeling Security Patterns as Contextual Goal Models

A security pattern captures proven security knowledge in a structured template with a number of sections, among which there are four essential sections *context*, *problem*, *force*, and *solution*. Apart from these, a security pattern normally also contains other sections, such as *Example, Implementation, Structure, Dynamics, Consequence, Known Uses, See Also*. Table 1 shows an example, which contains the four essential sections, with more information presented in [3].

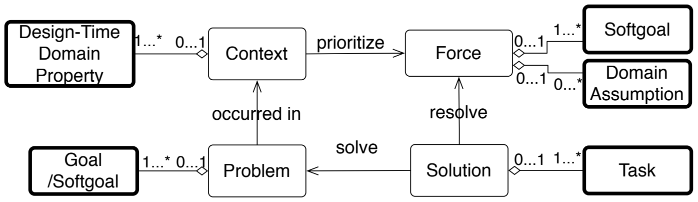


Fig. 1: Concept mappings between contextual goal models and security patterns

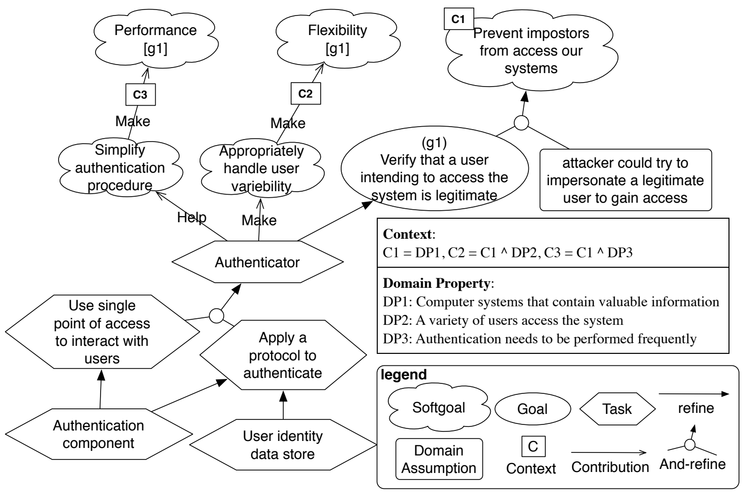
As claimed in section 2, we aim to analyze the context of security patterns. Towards this goal, we extend our previous modeling language with the following concepts. A *Domain Property* is a fact about the world, while a *Design-Time Domain Property* is a domain property that can be determined at design time. For example, *Computer systems on a local network connected to the Internet* is a design-time domain property, and we can verify this fact during design time according to the designed system infrastructure. For another example, *The number* *of users increases significantly* is a domain property, but not a design-time domain property, as it cannot be verified until the system is running. Because the security pattern analysis is carried out in the system design phase, we only capture and analyze *Design-Time Domain Properties*, and use this concept to describe the context of security patterns. A particular context could be an aggregation of domain properties in any complexity, typically, via an *and/or* operator.

Based on the above extension, we propose a mapping between the four essential concepts of the security pattern and the concepts of the contextual goal model, which is presented in Fig. 1. We describe the mapping in detail as below, where the definition of each security pattern concept is introduced first.

- **Context**: is the circumstance in which the security problem occurs and is being solved, and it also determines the importance of the forces. We model the context with an aggregation of *Domain Property*, via *and/or* operator.
- **Problem**: is a description of a situation where stakeholders do not have a solution. We use one or several *Goals* or *Softgoals* to capture stakeholders' needs concerning such a problem.
- **Force**: are considerations, often contradictory, which have to be taken into account when choosing a solution to a problem. These considerations are often related to non-functional requirements, such as performance and cost. We model such forces as *Softgoal*s. Moreover, some forces describe designer's assumptions, which guarantee the security problems can be correctly tackled by the security patterns, such as *Network firewalls cannot filter out harmful message*. We use *Domain Assumption* to model such forces.
- **Solution**: describes detailed actions that are carried out by a security pattern. We model them as *Task*s, which the system-to-be performs to satisfy its requirements.

Although the mappings we proposed above conceptually match concepts in two sides, our practical experiences reveal a number of difficulties during the transformations. One important reason for these difficulties is that security patterns are sometimes documented in different ways. For example, some patterns omit the *Force* section, such as the pattern *Secure Adapter*. To facilitate this transformation, we provide complementary guidelines: 1) Some patterns not only present forces in the *Force* section, but also in the *Consequence* section. 2) Apart from the general context specified in the *Context* section, the problems, forces, and solutions may involve particular contexts. For example, as shown in Fig. 2, the authenticator pattern can contribute to the softgoal *Flexibility* only under the context (*C*2) that *a variety of users access the system*. 3) Some solutions are described in a very abstract manner, which should be further detailed by reviewing the *Implementation* section.

Based on the proposed mapping and guidelines, we transform the textual security pattern shown in Table 1 into a contextual goal model, shown in Fig. 2. Note that we adopt the method proposed in [1] to model *Context* in goal models using their semantics. For example, the goal *Prevent imposters from access our systems* is activated, if and only if context C1 holds. The context C1 is described as *DP*1, which is extracted from the *Context* section of Table 1.



Performance [g1]
Flexibility [g1]
C1
Prevent impostors from access our systems
C3
C2
Make
Make
Simplify authentication procedure
Appropriately handle user variebility
(g1) Verify that a user intending to access the system is legitimate
attacker could try to impersonate a legitimate user to gain access
Help
Make
Authenticator
Context:
C1 = DP1, C2 = C1 ^ DP2, C3 = C1 ^ DP3
Domain Property:
DP1: Computer systems that contain valuable information
DP2: A variety of users access the system
DP3: Authentication needs to be performed frequently
Use single point of access to interact with users
Apply a protocol to authenticate
Authentication component
User identity data store
legend
Softgoal
Goal
Task
refine
Domain Assumption
C
Context
Contribution
And-refine

Fig. 2: Transformation example

4 Analyzing Security Patterns

Once obtaining a set of security patterns modeled in contextual goal models, we propose a systematic analysis procedure to analyze security patterns, which can be further integrated into our three-layer framework to enhance the security requirements operationalization analysis.

– **Pattern Candidate Generation.** By using our three-layer security requirements analysis [4], we can automatically derive a number of security pattern candidates with regard to a critical security goal.
– **Context Verification.** The context of each security pattern candidate is checked against the current system context. Our three-layer requirements models can be used for this verification, because they cover system-related design facts, ranging from the business layer to the physical layer. Because contexts may not all be explicitly presented, this step will be done in an interactive manner with stakeholders. If the context of a security pattern does not hold, we should deactivate its corresponding part in the goal model pattern as specified in [1], and further propagate the deactivation through the *refine* and *and-refine* links. For example, if the context *C*1 does not hold in Fig. 2, this pattern will not apply, as the *C*1 is tagged on the root goal.
– **Pattern Selection.** After above analysis, if there is more than one candidate, we can apply certain selection algorithm to select the best security pattern w.r.t its contributions to forces. For example, a NFR-based approach has been proposed for this selection in [2].
– **Pattern Application.** If a security pattern is chosen to be implemented, we insert the whole goal model pattern into the three-layer requirements models. Because the solution has been refined to the detailed and layer-specific tasks, such as *Authentication component* in Fig. 2, the analyst does not need to further refine the leaf tasks but only needs to assign them to corresponding actors in the three-layer requirements models.

5 Related Work

Mouratidis et al. extend Secure Tropos methodology with the use of security patterns [5]. They model security patterns in agent-oriented goal models, which capture both detailed security components and their interactions. However, their approach does not consider contexts of security patterns during their selections. Araujo and Weiss [2] propose to apply the Non-Functional Requirement (NFR) framework as a complementary representation for security patterns, which helps to analyze the tradeoffs between forces. Although they also do not consider the influences of context on the NFR models, this work could be employed in our work after analyzing and adding the contextual issues to the model. Ali et al. [1] propose a contextual goal modeling language, which explicitly models *Monitorable Context* in goal models and further provides related reasoning algorithms. This work focuses on runtime requirements analysis, while our framework analyzes requirements at design time. We follow their method to model contexts, but redefine some of their concepts to fit our needs.

6 Conclusion

In this paper, we propose to model security patterns in terms of contextual goal models to facilitate the selection and application of security patterns. Therefore, we define conceptual mappings and provide guidelines to assist the modeling of security patterns. Furthermore, we propose a systematic process to analyze security patterns, which can enhance our previous work. Currently, we have constructed contextual goal models for 6 security patterns in [3], serving as starting examples. In the future, we plan to transform a significant number of security patterns to support application of our security framework to a realistic case study. As we briefly discussed in the Sec. 3, security pattern knowledge does not always strictly fall within the corresponding sections. Along with the above transformation, we also have the aim of improving existing security pattern documentations. Finally, we plan to develop a tool to model security patterns and semi-automate analysis, and further integrate this work into our three-layer security analysis to better design secure systems.

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

References

1. R. Ali, F. Dalpiaz, and P. Giorgini. A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering*, 15(4):439–458, 2010.
2. I. Araujo and M. Weiss. Linking patterns and non-functional requirements. In *Proceedings of the Ninth Conference on Pattern Language of Programs (PLOP 2002), September 8-12, 2002*, 2002.
3. E. Fernandez-Buglioni. *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons, 2013.
4. T. Li and J. Horkoff. Dealing with security requirements for socio-technical systems: A holistic approach. In *the 26th International Conference on Advanced Information Systems Engineering (CAiSE'14)*. Accepted, 2014.
5. H. Mouratidis, M. Weiss, and P. Giorgini. Modeling secure systems using an agent-oriented approach and security patterns. *International Journal of Software Engineering and Knowledge Engineering*, 16(3):471, 2006.