Dealing with Security Requirements for Socio-Technical Systems: A Holistic Approach

Tong Li and Jennifer Horkoff

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

Abstract. Security has been a growing concern for most large organizations, especially financial and government institutions, as security breaches in the socio-technical systems they depend on are costing billions. A major reason for these breaches is that socio-technical systems are designed in a piecemeal rather than a holistic fashion that leaves parts of a system vulnerable. To tackle this problem, we propose a three-layer security analysis framework for socio-technical systems involving business processes, applications and physical infrastructure. In our proposal, global security requirements lead to local security requirements that cut across layers and upper-layer security analysis influences analysis at lower layers. Moreover, we propose a set of analytical methods and a systematic process that together drive security requirements analysis throughout the three-layer framework. Our proposal supports analysts who are not security experts by defining transformation rules that guide the corresponding analysis. We use a smart grid example to illustrate our approach.

Keywords: Security Requirements, Goal Model, Multilayer, Socio-Technical System, Security Pattern.

1 Introduction

Like all non-functional requirements, security requirements have a global influence over the design of a socio-technical system. Socio-technical systems (STSs) are organizational systems consisting of people, business processes, software applications, and hardware components. Such systems often include a rich physical infrastructure consisting of not only computers, but also buildings, cable networks and the like. Due to their ever-increasing complexity, STSs have been experiencing a growing number of security breaches [5], caused by security flaws and vulnerabilities.

A common theme for many of these breaches is that security solutions are not designed in a holistic fashion. Rather, they are dealt with in a piecemeal fashion, by different analysts and at different times, using different analysis techniques. For example, Mouratidis [13] and Liu [11] analyze security issues at organizational level; Herrmann [9] analyzes security requirements in business process level; Lamsweerde investigate security requirements for software [21]. This leads to security gaps and vulnerabilities for parts of a STS, while others

M. Jarke et al. (Eds.): CAiSE 2014, LNCS 8484, pp. 285–300, 2014.

may be costly and over-protected. For example, when designing an encryption function for a smart meter system, a designer may focus only on the software (application layer). In this case, the software can implement encryption by calling functions implemented by an external hardware chip. However, with this design alternative, calling the external functions means that the software first sends unencrypted text to the chip, creating a vulnerability which can be exploited by non-authorized people (business layer) via bus-snooping (physical layer) [1]. By focusing only on the software, vulnerabilities from the physical layer and business layer perspectives are missed.

To tackle this problem, we propose a holistic approach to security engineering where STSs consist of three layers: a business layer, a (software) application layer, and a physical infrastructure layer. Within this framework, each layer focuses on particular concerns and has its own requirements and specifications, which are captured by goal-oriented requirements modeling language. In particular, specifications in one layer dictate requirements in lower layers. In this manner, the security requirements analysis carried out in one layer seamlessly leads to the analysis in the next layer down. Thus, security requirements derived in different layers can cooperate properly to deliver security to systems. Go back to the aforementioned encryption example, if a holistic view is taken, alternative security treatments are identified: 1) apply software-based encryption, which avoid hardware access issues; 2) apply hardware-based encryption, as well as additional protections on corresponding hardware. In this way, security mechanisms applied in different layers are coordinated, and the aforementioned vulnerability can be avoid.

Based on this framework, a systematic process is provided to drive security analysis both within one layer and across layers. To support analysts without much security knowledge, we propose a set of analytical methods and corresponding transformation rules to facilitate the security analysis process. This security analysis framework is particularly designed for existing systems that have a determined functional design. Our approach takes a number of high-level security requirements as input, analyzes their influences over three layers, and produces holistic security treatments that consist of coordinated security mechanisms in different layers.

In the reminder of this paper, we first describe a smart grid example in Section 2, which is used to illustrate our approach throughout the paper. Next, in Section 3 we introduce our research baseline on requirements and specification models, and security requirements analysis. Section 4 presents the three-layer security analysis framework, while Section 5 describes a set of security requirement analysis methods and a systematic analysis process. Section 6 compares our proposal to other work, and finally in Section 7 we conclude the whole paper and discuss future work.

2 Motivating Example

In this section, we introduce a smart grid example, which leverages information and communication technologies to enable two-way communications between

customers and energy providers. This example involves a number of scenarios, and we exclusively focus on a *real-time pricing* scenario. In this scenario, the service provider periodically collects customer's energy consumption data, based on which they can create new prices to balance loads on the power grid. A business layer requirement model for this scenario is shown in Fig. 1. We will introduce details of the goal-oriented modeling language in Section 4.

Because this system involves a wide scope of artifacts, which vary from business processes to physical devices, it is difficult to provide a cost-benefit security treatment to protect the whole system from damages. As reported by National Vulnerability Database, on average, 15 new vulnerabilities of the Supervisory Control and Data Acquisition system (a major control system used in power grid systems) are publicly disclosed each day. Not surprisingly, the presence of these vulnerabilities leads to many attacks on smart grid systems [5].

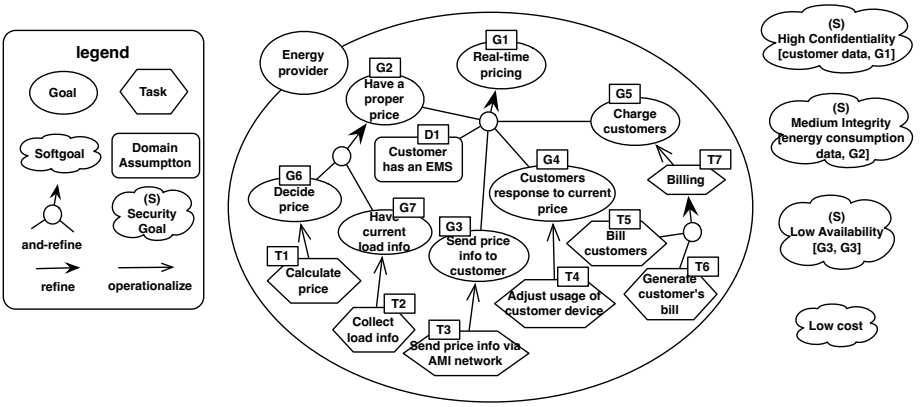

Fig. 1. High-level requirements of real-time pricing scenario

3 Baseline

In this section, we introduce existing work, used as the baseline of our research. We first introduce the requirements problem, which specifies fundamental tasks that need to be addressed during requirements analysis. Then, we describe several requirements modeling language [10,22,6,2], which are intended to capture requirements or tackle the requirements problem.

Requirements Problem. Zave and Jackson [23] define a Requirements Engineering ontology to specify what is the requirements problem. This definition consists of three concepts: a *Requirement* is an optative property that specifies stakeholder's needs on the system-to-be; a *Domain Assumption* is an indicative property that is relevant to the system; a *Specification* is an optative property, which is able to be directly implemented by the system-to-be. Based on these three basic concepts, they define the requirements problem amounts to finding a set of specifications

S, which can satisfy all system requirements R under domain assumptions K. Thus, the requirements problem is represented as $K, S \vdash R$.

Requirements Modeling Language. Jureta et al. [10] propose a goal-oriented requirements modeling language *Techne*, which includes all related concepts for addressing the requirements problem as defined by Zave and Jackson. In addition, it is able to model stakeholder's priority over different requirements, based on which, the best solution can be obtained amongst candidate solutions. Yu [22] proposes the *i** framework for modeling organizational environments and system requirements. Specifically, it captures relationships among social actors via dependency relations. Chung adopts NFR to analyze security requirements [2]. In this work, a security requirement is represented as a *security goal*, which is specified in terms of *sort* and *parameter*. Giorgini et al. [6] model trusts relations between social actors in order to analyze social security issues. In this work, we base our three-layer framework on a combination of above approaches to model both functional and non-functional requirements (including security requirements), as well as social interactions. Particularly, we use the concepts *actor, goal, softgoal, quality constraint, task, and domain assumption* and the relations *refine, preferredTo, dependency, contribution, and trust*, provided by those approaches. Fig. 3 shows how we combine these concepts and relations in our conceptual model.

4 Three-Layer Security Analysis Framework

Stakeholder's global security needs, which are captured as non-functional requirements, influence designs in all parts of the system. We propose to structure a system into three layers, and analyze security issues in each layer from a holistic viewpoint.

4.1 Three-Layer Structure

In this work, we have focused on three particular layers, which have received much attention from the security community. As shown in Fig. 2, at the most abstract layer, we consider the business layer, which highlights social dependencies, trusts, and business processes. At the next layer of abstraction, we consider software applications and their related IT infrastructures. Finally, we consider the physical infrastructure layer, which focuses on deployments of software applications and placements of devices.

As an essential part of the three-layer structure, we propose to analyze the requirements problem for each layer respectively. Each layer has its own requirements R, which are operationalized into proper specifications S under corresponding domain assumptions K. As shown in the left part of Fig. 2, we apply goal-oriented modeling to each of the three layers with the aim of analyzing their requirements problems respectively.

We base our three-layer security analysis approach on the proposed three-layer structure, which is shown in the right part of Fig. 2. Our approach starts

with high-level stakeholder's security requirements; and then analyzes them throughout three layers with regard to layer-specific goal models; and finally generates a set of alternative global security treatments.

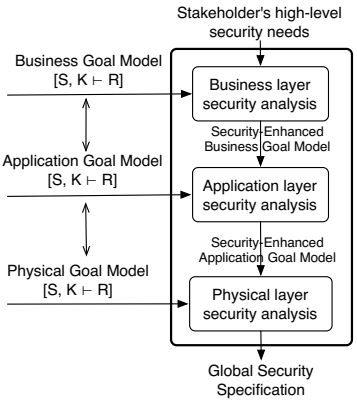

Fig. 2. Framework overview

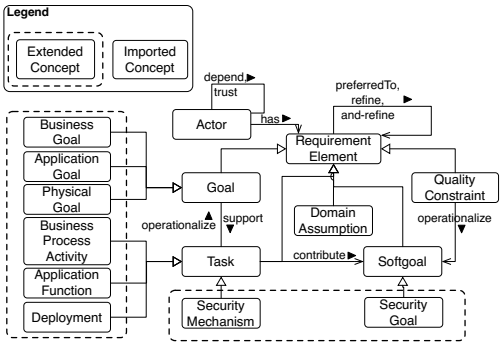

Fig. 3. Conceptual model of the Goal Model within the three-layer framework

4.2 Three-Layer Conceptual Models

In this section, we specify conceptual models that we use for modeling and analyzing the three-layer architecture. Apart from concepts and relations we adopt from existing approaches, mentioned in Section 3, we further extend and make use of new concepts. Fig. 3 shows an overview of conceptual model of the three-layer framework, where the newly introduced concepts are highlighted in the dashed rectangles.

Extended Requirement Concepts and Relations. As we build goal models for different layers capturing different concerns, we specialize *Goal* into layer-specific goals that focus on a particular aspect. *Business Goal* represents stakeholder's high-level requirements for his business. *Application Goal* represents stakeholder's requirements regarding software applications that he uses to perform related business activities. *Physical Goal* represents stakeholder's requirements on physical devices and facilities that support execution of software. Accordingly, we assign *Task* in different layers with operational definitions. In the business layer, a task is a *Business Process Activity*; in the application layer, an *Application Function* is deemed as a task; and in the physical layer, a task is specialized into a *Deployment* action.

Apart from that, a number of relations are also proposed. *Operationalize* is a relation that presents how a goal/softgoal is operationalized into a task/quality constraint. This relation emphasizes the relationship between requirements and specifications, and indicates when a stakeholder's requirements are translated into operational specifications. For example, the business goal *Have current load*

info shown in Fig. 1, which is desired by the *Energy Provider*, is operationalized into the task *Collect load info* in the same layer, which can accomplish this goal. *Support* is a cross-layer relation, which specifies a task designed in one layer is supported by requirements in the next layer down. Fig. 9 contains examples of this relation.

Extended Security Requirement Concepts. Chung [2] leverages non-functional requirements analysis to deal with security requirements, which are represented as *security goals*. Each security goal consists of one *sort* and one or more *parameters*. In our framework, we extend *security goals* to express more detailed security requirements, and introduce *security mechanisms* to represent security solutions.

Security Goal represents stakeholder's security needs with regard to asset and interval. We define a security goal as a specialization of softgoal, which particularly focuses on security issues. A security goal is specified in the format: *<importance><security attribute>[<asset>, <interval>]*. Take the security goal *Medium Integrity [energy consumption data, G2]* (in Fig. 1) as an example, its four dimensions together describe a security requirement "protecting integrity of energy consumption data during the execution interval of G2 to a medium degree".

- *Security Attribute* specifies a characteristic of security. Particularly, we adopt security attributes use in [17,4], which is shown in Fig. 5. The security attributes we consider in our work constitute a minimum set, which serves as a starting point and can be extended in the future.
- *Asset* is anything that has value to an organization, such as data, service. Fig. 4 shows an overview of all the types of assets that we have considered in our framework, as well as the interrelationships among them. Normally different assets are concerned in different layers. For example, we only consider *Service* and *Data* as assets in the business layer.
- *Interval* of a security goal indicates within which temporal interval the security goal is concerned. In this work, an interval is specified in terms of the execution period of a goal or task.
- *Importance* of a security goal indicates to which degree stakeholders want the security goal to be satisfied. We consider the value of importance within an enumeration {*very low, low, medium, high, very high*}.

Security Mechanism is a concrete mechanism provided by the "system-to-be" in order to achieve one or more security goals. We define the security mechanism

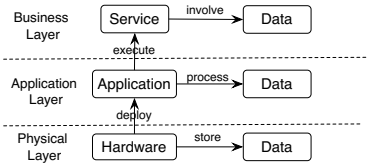

Fig. 4. Overview of assets

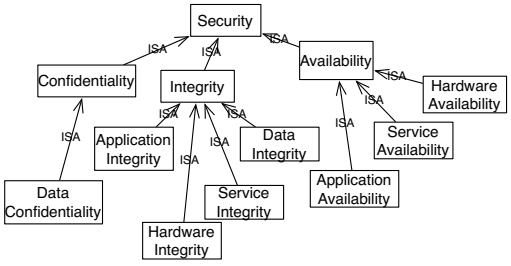

Fig. 5. Hierarchy of security attributes

as a specialization of task in goal model, which contributes to security goals and satisfies them. Thus, in our framework, security mechanisms are parts of specifications, and also influences requirements in its lower layers.

5 Security Analysis Methods

In this section, we propose a systematic process and a set of security analysis methods to guide security analysis both within one layer and across layers. Fig. 6 shows an overview of the analysis process, which starts from security analysis in the business layer and follows a top-down manner to propagate influences of security analysis in one layer to lower layers. Within one single layer, we refine and simplify security goals to identify concrete and critical ones, which are then operationalized into possible security mechanisms that are left to security analysts to select. After security analysis has been done for all layers, security treatments applied in each layer are synthesized to generate holistic security treatments.

We propose security analysis methods and corresponding transformation rules to guide the aforementioned analysis steps, which have been implemented using Datalog rules. We developed a prototype for a CASE tool, which automates some of the analysis steps as indicated in Fig. 6. Due to space limitation, we only describe and illustrate a small part of the transformation rules. A full list of the 23 transformation rules is available online [1]. In the reminder of this section, we describe details the proposed security analysis methods in subsections, each of which support one or several analysis task shown in Fig. 6.

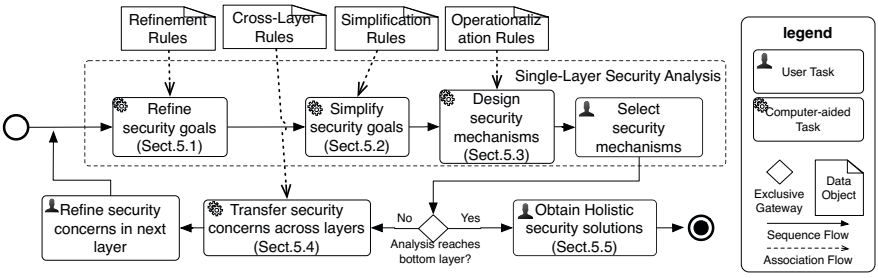

Fig. 6. An overview of the three-layer security requirements analysis process

5.1 Refinement Methods

A coarse-grained security goal is normally more difficult to analyze and operationalize than a fine-grained one, as it may be too abstract to be satisfied by specific security mechanisms. Thus, it is advisable for an analyst to refine a security goal till he obtains satisfiable ones. A security goal can be refined along

[1] http://goo.gl/Pd0TGw

any of its dimensions (security attributes, asset, or interval), i.e. there are three refinement methods. Fig. 7 shows an example of security goal refinements for security goal *Medium Integrity [energy consumption data, G2]* (Fig. 1). Note that a refinement process can be flexible in the sense that different refinement methods can be applied in any sequence and to any extent. Given the reference models that are shown in the left part of Fig. 7, the example presents only one possible way to refine the goal.

- **Security attributes-based refinement:** refining security goals via security attributes helps the security analysis to cover all possible aspects of security. According to the hierarchy of security attributes shown in Fig. 5, a security goal that talks about a high-level security attribute can be refined into several sub-security goals that talks about corresponding low-level security attributes. For example, in Fig. 7, the security goal *Medium Integrity [energy consumption data, G2]* is refined into four sub-security goals.
- **Asset-based refinement:** refinement of security goals can also be done by refining assets via *part-of* relations, which propagates a security goal on an asset to all its components. The *part-of* relation is an abstract one, which can be specialized into particular types of *part-of* relation of different conceptual models, such as data schema, software architecture model etc. For example, the security goal *Medium Data Integrity [energy consumption data, G2]* (Fig. 7) is refined according to the *part-of* relations among energy consumption data, water consumption data, and electricity data.
- **Interval-based refinement:** because an interval specifies the temporal period, for which a security goal is concerned, the security analyst can put more detailed constraints on a particular time intervals by refining a long interval into smaller ones. Here, we use the execution periods of requirement goal/tasks to represent intervals of time. Thus the interval-based security goal refinements are carried out according to the refinement of system functionality in the requirements models. For example, the four leaf nodes (in Fig. 7) are refined according to the functional requirements model in Fig. 1.

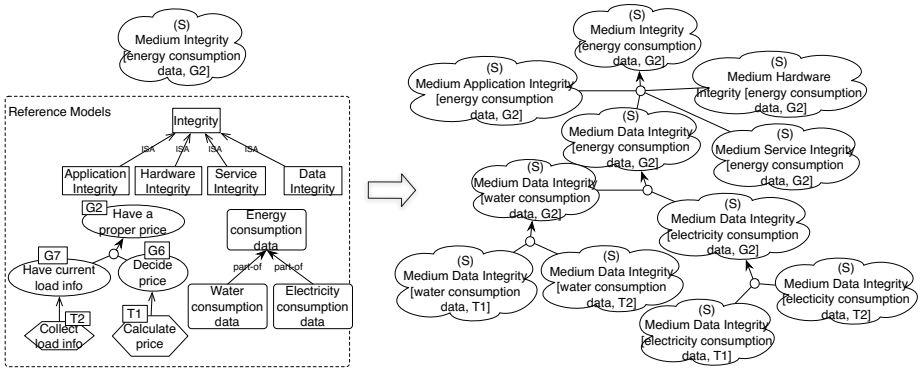

Fig. 7. Security goals refinements

5.2 Simplification Methods

When dealing with a large number of security goals, analysts may not have enough time to go through each of them to determine which is critical and requires further treatments. Especially as the refinement methods, which are intended to cover every potential facet of security goals, easily result in many detailed security goals. To release analysts from scrutinizing all security goals, we introduce simplification methods, which identify critical security goals that need to be treated and exclude others.

In order to determine the criticality of security goal, we consider two particular factors: **applicability** and **risk level**. The applicability specifies whether a security goal is sensible with regard to the content of the four dimensions of a security goal. The risk level identifies to which extent the satisfaction of a security goal is threatened. The criticality of a security goal is determined by considering: 1) If a security goal is applicable and its risk level is either high or very high, then we treat this security goal as a critical one, which is highlighted with a character ''C''. 2) All other security goals are deemed as non-critical and will be removed from following security analysis. It is worth noting that this criticality analysis can be adjusted depending on analyst time and the domain. For example, if analyst time allows, a security goal, which is at the medium risk level, could be adjusted as critical security goal.

For the analysis of applicability, we consider not only related requirements and simple specifications, but also detailed specifications, i.e. design information. For instance, how business activities are arranged, how application components interact with others, and where physical devices are placed. Based on this information, we propose layer-specific inference rules to determine the applicability of a security goal in one layer. For example, the security goal *Medium Data Integrity [water consumption data, T1]* (Fig. 7) is not suitable, because the asset *water consumption data* would not be changed during the execution of *T1*. Due to space limitation, in each layer we only present one rule as an example, which are shown in Table 1.

For the risk level analysis, we carry out a trust-based approach, which consider the trust between the owner of a security goal and the actor that potentially

Table 1. Rules for determining applicability of security goal

Rule	Rationale
BUS.A.1	If a business process activity takes a data asset as input, the confidentiality of this asset might be impaired within this activity. Thus, corresponding security goals are identified as applicable.
APP.S.1	If an application component is called by another component for a data asset, the integrity and confidentiality of that data asset may be impaired during functioning of that component. Thus, corresponding security goals are identified as applicable.
INF.S.1	If a hardware device stores a data asset, the integrity and confidentiality of that data asset may be impaired during deployment task of that device. Thus, corresponding security goals are identified as applicable.

impairs the security goal. For example, the maintainer of a smart meter may impair the integrity of that meter, if a customer owns a security goal that aims to protect the integrity of the smart meter, then our analysis considers to which extent the customer trusts the maintainer. Table 2 represents how we infer the risk level of security goals with regard to the trust level and the importance of security goals. Note that the letter L, M, H, V stands for low, medium, high, and very high level of risk respectively.

Table 2. Risk level evaluation matrix

Trust \ Importance	very low	low	medium	high	very high
very bad	H	H	V	V	V
bad	M	H	H	V	V
neutral	L	M	H	H	V
good	L	L	M	H	H
very good	L	L	L	M	H

5.3 Operationalization Methods

To bridge the gap between security requirements and security specifications within an individual layer, we propose operationalization methods which generate possible security mechanisms that could satisfy critical security goals. As security mechanism analysis and design requires additional security knowledge, which is normally not easy to obtain in reality, we exploit the power of security patterns to reuse security knowledge that tackles known security problems. Particularly, we survey existing work on security patterns [17,8,19], and extract the parts of them that are suitable for our security analysis framework. Note that the selection of security patterns is not intended to be exhaustive, and may evolve over time.

A security pattern consists of a security attribute and a security mechanism that is supposed to satisfy that security attribute. Each security mechanism can contribute to one or several security attributes, and we use *Make* and *Help* links to represent their contributions. For example, one security pattern shown in Fig. 8 is *Auditing* has a *Make* contribution to *Service Integrity*. According to these security patterns, when operationalizing a critical security goal, we identify security mechanisms that contribute to the security attribute of the security goal. It is worth noting that security patterns, shown in the Fig. 8, may also have either positive or negative influences on other non-functional goals, such as *Time, Cost*, which are documented in the specification of the security patterns.

5.4 Cross-Layer Analysis Methods

After finishing single-layer security analysis in one layer, indicated in Fig. 6, we analyze its influences on lower-layers. In our framework, the influences are

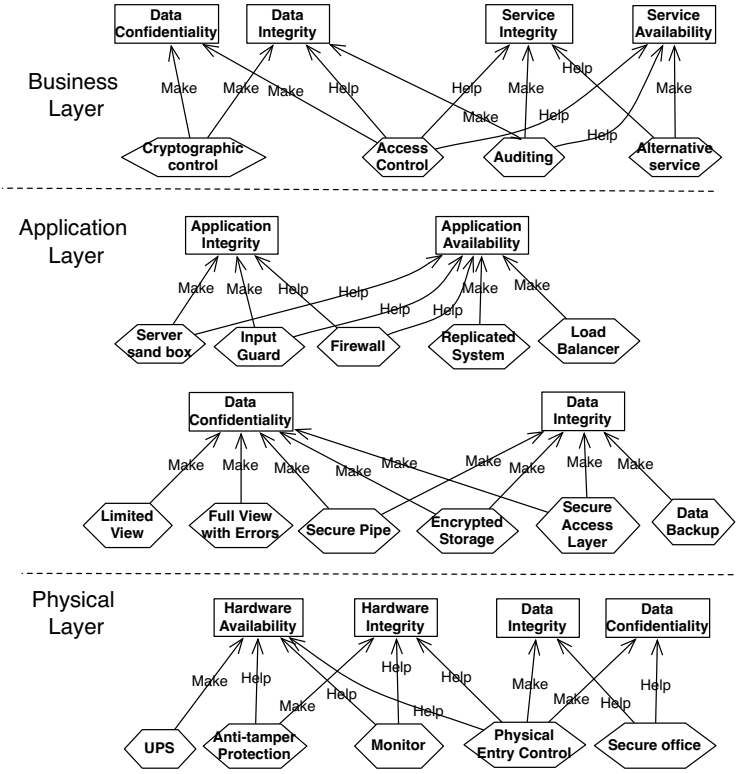

Fig. 8. Applicable security patterns in three layers

reflected in two ways, which require different analysis methods. Accordingly, a number of inference rules are proposed to automate the cross-layer transformations, parts of which are shown in Table 3.

Firstly, each of the security mechanisms should be transformed into at least one goal in the lower-layer goal model. Because security mechanisms are critical for satisfying security goals, additional security goals are derived to ensure correct implementations of the corresponding functional goals. Fig. 9(a) shows an example of this transformation. The security mechanism *Encryption* is transformed into a function goal of *smart meter application*, because that application is supposed to execute the *encryption* activity. Apart from this functional goal, the transformation also introduces two security goals, which concerns *application integrity* and *application availability* of the smart meter application during the execution of *encrypt data*. Note that the importance of these two new security goals is *Medium*, which is the same with the corresponding security goal in the business layer.

Secondly, if a security goal has not been fully satisfied in one layer, this security goal will be refined into security goals in next layer down according to the newly available information in that layer. Fig. 9(b) shows an example of

this transformation. The security goal *Medium Data Integrity [energy consumption data, Measure energy data]* is not treated in the business layer. Considering the asset of this security goal is processed by the *smart meter application* in the application layer, the transformation rule *BUStoAPP.2* is applied, which results in two sub-security goals.

5.5 Global Security Analysis

After security analysis has been done in each layer, we can derive alternative global security treatments by synthesizing security analysis results of each layer. Each global security treatment may consist of security mechanisms from one or several layers. Take a snippet of the three-layer security analysis results of the smart grid as an example, which is shown in Fig. 10.

In the business layer, the critical security goal can be either operationalized as *Auditing* or left to the next layer. In the first case, the *Auditing* security mechanism is transformed into a functional goal and two corresponding security goals. As the two security goals are identified as non-critical after refinement and simplification analysis, no further security treatments are required. Thus, we derive the first global security treatment. In the second case, the security goal is refined into two sub-security goals in the application layer, one of which is identified as critical. This critical security goal can be operationalized as either *Firewall* or *Input Guard*, each of which starts a new alternative. Apart from the operationalization, the security goal can also be left to next layer,

Table 3. Rules for cross-layer security goal refinements

Rule	Rationale
BUStoAPP.1	If a security goal concerns a service asset, which is supported by an application component, then the security goal introduces security goals that concern the integrity and availability of that application.
BUStoAPP.2	If an untreated security goal that concerns data confidentiality or data integrity targeting at a business process activity, which is supported by an application in the application layer, then this security goal will be refined to sub-goals that concern corresponding applications.

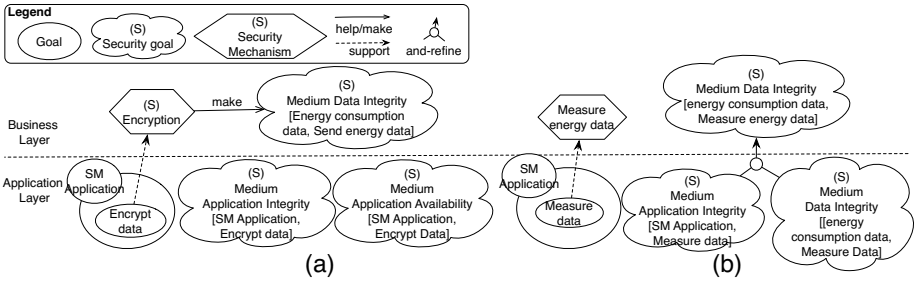

Fig. 9. Examples of security transformation

which further introduces another two alternatives. Finally, we derive 5 global security treatments in this example. Among these alternatives, the *Alt.3* contains a security mechanism from only one layer, i.e. *Input Guard*; while the *Alt.2* contains two security mechanisms from two different layers, i.e. *Firewall*, *Anti-tamper protection*.

6 Related Work

NFR-Based Requirements Analysis. Chung proposes to treat security requirements as a class of NFRs, and apply a process-oriented approach to analyze security requirements [2]. In a subsequent work, Chung and Supakkul integrate NFRs with FRs in the UML use case model [3], which enable NFRs to be refined through functional requirement models. Another complementary work done by Gross and Yu propose to connect NFRs to designs via patterns [7]. However, all of these NFR-based approaches mainly focus on information system analysis, and lack of supporting requirements analysis in the business layer and the physical layer.

Security Requirement Analysis. A large number of security requirement analysis approaches have been proposed over last two decades. Most of these approaches focus on analyzing security requirements with regard to a particular aspect of information system. There are approaches that focus on the social and

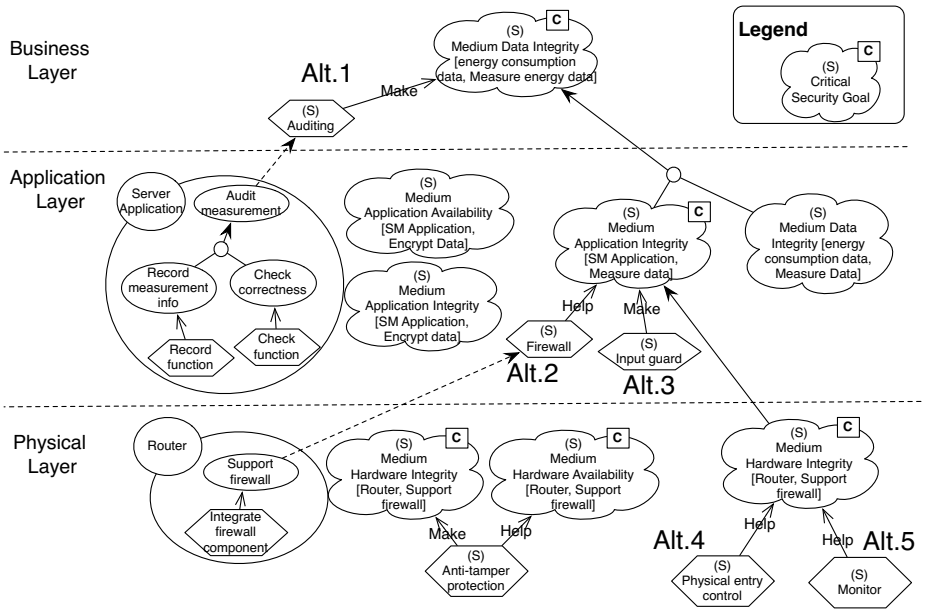

Fig. 10. A snippet of three-layer security analysis result of the smart grid example

organizational aspect. Mouratidis et al. [13] capture security intentions of stakeholders and interdependence among stakeholders; Giorgini et al. [6] investigate social relationships by integrating trust and ownership into security analysis; Liu et al. [11] analyze organizational risks by considering dependencies among social actors. Another branch of work deals with security requirements for business process. Rodríguez et al. [15] propose an extention of UML activity diagram to model security requirements within business process model, as well as a method that guides construction of the secure business process models. Herrmann et al. [9] propose a systematic process to elicit and analyze security requirements from business processes models. Most work is dedicated to analyzing security requirements of software, such as Attack Tree [18], Misuse case [20], and obstacle analysis [21]. All of these approaches are complementary to our approaches, as each of them could fit into one layer of our framework.

From Security Requirements to System Design. A number of approaches have been proposed to transform security requirements captured in the high-abstraction level to the security design in the low-abstraction level in order to maintain security requirements throughout the whole life-cycle of system development. Mouratidis and Jürjen [14] translate security requirements into design by combining Security Tropos with UMLsec. Menzel et al. [12] propose an approach that transfers security requirements, which are captured at the business process layer, to security configuration for service-based systems by using patterns. Similarly, Rodríguez [16] et al. apply the MDA technique to transform secure business process model into analysis class diagram and use case diagram. The above approaches focus on maintaining security requirements identified in the early stage during later design stages. Different from their work, we propose to analyze security requirements in different layers from a holistic viewpoint and capture influences between security requirements in different layers.

7 Conclusions and Future Work

In this paper, we propose a holistic approach to analyze security requirements for STSs. Our approach consists of a three-layer conceptual model, a systematic analysis process, a number of security analysis methods and transformation rules. Given high-level security requirements, this approach could continuously refine and propagate them into different layers of socio-technical systems. The approach focuses on capturing the influence of upper-layer designs on the requirements of lower layers, thus avoiding a piece-meal treatment of security.

As we use goal modeling for security requirement analysis, we are bound to deal with complexity of such models. Compared to other goal-based approaches that model a system as a whole, ours structures STSs into three layers, each of which is modeled separately and connected via a clearly defined conceptual model. Thus, our approach contributes to reduce the scalability issues in individual model. In addition, we develop a prototype tool to facilitate our analysis, which supports graphic modeling and automates inferences over the transformation rules. However, our approach has limitations on its evaluation.

So far, our approach is only applied to a single scenario of the Smart Grid case, which is an illustrative example rather than a practical evaluation.

In the future, we plan to extend our framework by incorporating real security regulations and laws, such as ISO standards, in order to provide more practical and grounded security analysis. Moreover, the security patterns we leveraged during security goal operationalization should be updated in light of recent advances in the field [19] synchronized with the cutting edge of that field. Finally, with the help of the prototype, we intend to apply our approach to a practical case study that has a reasonable scale to evaluate and further improve our work.

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

References

1. Carpenter, M., Goodspeed, T., Singletary, B., Skoudis, E., Wright, J.: Advanced metering infrastructure attack methodology. InGuardians White Paper (2009)
2. Chung, L.: Dealing with security requirements during the development of information systems. In: Rolland, C., Cauvet, C., Bodart, F. (eds.) CAiSE 1993. LNCS, vol. 685, pp. 234–251. Springer, Heidelberg (1993)
3. Chung, L., Supakkul, S.: Representing nfrs and frs: A goal-oriented and use case driven approach. In: Dosch, W., Lee, R.Y., Wu, C. (eds.) SERA 2004. LNCS, vol. 3647, pp. 29–41. Springer, Heidelberg (2006)
4. Firesmith, D.: Specifying reusable security requirements. Journal of Object Technology 3(1), 61–75 (2004)
5. Flick, T., Morehouse, J.: Securing the smart grid: next generation power grid security. Elsevier (2010)
6. Giorgini, P., Massacci, F., Zannone, N.: Security and trust requirements engineering. In: Aldini, A., Gorrieri, R., Martinelli, F. (eds.) FOSAD 2005. LNCS, vol. 3655, pp. 237–272. Springer, Heidelberg (2005)
7. Gross, D., Yu, E.: From non-functional requirements to design through patterns. Requirements Engineering 6(1), 18–36 (2001)
8. Hafiz, M., Adamczyk, P., Johnson, R.E.: Organizing security patterns. IEEE Software 24(4), 52–60 (2007)
9. Herrmann, P., Herrmann, G.: Security requirement analysis of business processes. Electronic Commerce Research 6(3-4), 305–335 (2006)
10. Jureta, I., Borgida, A., Ernst, N., Mylopoulos, J.: Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: Proc. of RE 2010, pp. 115–124 (2010)
11. Liu, L., Yu, E., Mylopoulos, J.: Security and privacy requirements analysis within a social setting. In: Proc. of RE 2003, Monterey, California, pp. 151–161 (2003)
12. Menzel, M., Thomas, I., Meinel, C.: Security requirements specification in service-oriented business process management. In: Proceedings of International Conference on Availability, Reliability and Security, ARES 2009, pp. 41–48. IEEE (2009)
13. Mouratidis, H., Giorgini, P.: A natural extension of tropos methodology for modelling security. In: Proc. of the Agent Oriented Methodologies Workshop (OOPSLA 2002). Citeseer, Seattle (2002)
14. Mouratidis, H., Jurjens, J.: From goal-driven security requirements engineering to secure design. International Journal of Intelligent System 25(8), 813–840 (2010)

15. Rodríguez, A., Fernández-Medina, E., Trujillo, J., Piattini, M.: Secure business process model specification through a uml 2.0 activity diagram profile. Decision Support Systems 51(3), 446–465 (2011)
16. de Rodríguez, G.I.G.R., Fernández-Medina, E., Piattini, M.: Semi-formal transformation of secure business processes into analysis class and use case models: An mda approach. Information and Software Technology 52(9), 945–971 (2010)
17. Scandariato, R., Yskout, K., Heyman, T., Joosen, W.: Architecting software with security patterns. Tech. rep., KU Leuven (2008)
18. Schneier, B.: Attack trees. Dr. Dobb's Journal 24(12), 21–29 (1999)
19. Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., Sommerlad, P.: Security Patterns: Integrating security and systems engineering. John Wiley & Sons, (2013)
20. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requirements Engineering 10(1), 34–44 (2005)
21. Van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. IEEE Transactions on Software Engineering 26(10), 978–1005 (2000)
22. Yu, E.: Towards modelling and reasoning support for early-phase requirements Engineering, pp. 226–235. IEEE Computer Soc. Press (1997)
23. Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. 6(1), 1–30 (1997)