Towards model-driven generation of secure workflow designs using patterns

Sotirios Liaskos[1], Ibrahim Jaouhar[1] and Shakil M. Khan[2]

[1] School of Information Technology, York University, Toronto, Canada
[2] Department of Computer Science, University of Regina, Regina, Canada

Abstract

Identifying and analyzing security requirements is considered to be an essential part of the requirements engineering process. Several techniques have been introduced for capturing and modeling such requirements. Once identified, security requirements must be translated into designs that suggest how domain actors can securely fulfill operational goals under varying contexts and risk assumptions. We propose a model-driven approach for automatically generating workflows that fulfill information exchange goals while complying with specified security requirements. Goal-oriented security requirements models, augmented with descriptions of contextual and threat assumptions, are combined with reusable domain-agnostic workflow patterns, which model ways for securely performing common information exchange tasks. The resulting combined models are used by an automated reasoner to generate workflows that meet security requirements. The approach and toolset can be used by requirements analysts to systematically generate secure designs from requirements, in a way that relies less on their technical expertise in security and more on established best practices embodied in the patterns. We present the basic components of our approach via examples and describe future research directions.

Keywords

Information Security, Goal Modeling, STS-ml, AI Planning

1. Introduction

Identifying and analyzing security requirements has long been considered to be an essential part of the requirements analysis process [1, 2]. A wealth of techniques for modeling such requirements have been proposed, aimed at allowing their precise description, documentation, and communication [3, 4, 5]. However, following capture of security requirements, a system design must be developed that, on one hand, satisfies stakeholder requirements completely and, on the other hand, complies with widely accepted security standards. Of particular interest are often cryptographic primitives that need to be used in specific ways in order to support the secure exchange of information – business documents, transactions, etc. Analysts, designers, and users with limited understanding of cryptography or access to experts, may be prone to erroneous uses of these primitives and/or misconceptions as to what they guarantee.

ER2023: Companion Proceedings of the 42nd International Conference on Conceptual Modeling: ER Forum, 7th SCME, Project Exhibitions, Posters and Demos, and Doctoral Consortium, November 06-09, 2023, Lisbon, Portugal

liaskos@yorku.ca (S. Liaskos); jaouhar@yorku.ca (I. Jaouhar); shakil.khan@uregina.ca (S. M. Khan)
https://www.yorku.ca/liaskos/ (S. Liaskos); http://www2.cs.uregina.ca/~skhan/ (S. M. Khan)
0000-0001-5625-5297 (S. Liaskos); 0000-0003-0140-3584 (S. M. Khan)

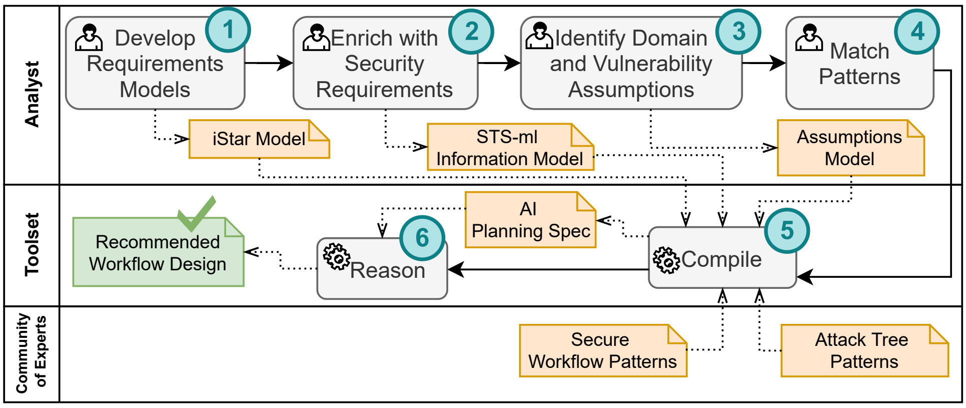

Figure 1: Overview of the proposed framework

We propose a model-based framework for systematically generating cryptography-enabled business workflows that domain actors can follow in order to fulfill their information exchange goals while also satisfying security requirements. The framework depends on the availability of expertly developed patterns that describe families of workflows for fulfilling common information exchange tasks. Security requirements are captured using a goal-oriented security requirements modeling language and analyzed via the development of attack trees that can compromise them, while the level of sophistication of the attacks to be defended against is captured though a set of vulnerability assumptions. The developed models are combined and translated in an automatable way into an Artificial Intelligence (AI) planning specification and an AI planner is used to automatically identify workflows that indicate security steps domain actors need to take while fulfilling information exchange goals. Using such a tool, designers can develop business workflows in a systematic, requirements-driven way, while respecting the properties and limitations of the security primitives their designs employ.

2. Approach and significance

Consider the example of a small-sized residential building contractor requesting quotes, submitting orders, and sending payments to various material suppliers. Typically, the contractor submits orders for materials, and once the materials have been delivered, an invoice is sent with a payment information, often in the form of a bank account number for sending a wire transfer. The contractor performs such interactions via email, which, however, introduces several threats, including the submission and fulfillment of unauthorized orders through spoofing, and invoice fraud, i.e., submission of a tampered invoice, where the payment information has been altered.

The proposed framework is aimed at allowing analysts or IT support professionals who work with clients such as the above contractor to systematically generate secure workflow designs to fulfill information exchange needs of their clients, without having in-depth knowledge of the involved security technologies. An overview can be seen in Figure 1. The analysts go through a sequence of steps for developing goal models describing the specific information exchange needs (Step 1) and enriching them with additional security requirements (Step 2), and assumptions describing the level of attack to be protected against (Step 3). Having captured these requirements, analysts match elements of the domain-specific models with generic secure workflow and attack tree patterns from a collection of such developed and supported by the community of experts (Step 4). The enriched models and the patterns are then transformed into a formal specification in a way that can be automated (Step 5). From this specification an automated reasoner generates a workflow (Step 6) which includes the steps necessary for the performance of the information exchange tasks, appropriately enriched with cryptographic tasks to attain the captured security goals. We offer more details about these steps below.

Steps 1 and 2: Goal Models and Security Requirements. We adopt iStar 2.0 [6] for capturing general functional and quality requirements and the social view of STS-ml (Socio-Technical Security modeling language [1]) for modeling the security aspects. In our example, the requirements model may include information exchange goals such as *Send Invoice to Contractor*, annotated with security service requirements such as *Confidentiality*, *Integrity*, or *Authentication.*

Step 3: Domain and vulnerability assumptions. Predicates are added representing two kinds of assumptions. *Domain assumptions* are practical aspects of the domain that may affect the applicability of security controls or attacks against them, such as *HasEmailAddress(contractor, supplier)* or *DigitalFile(invoice). Vulnerability assumptions* represent worst-case states of affairs in which possible attacks have been successfully carried out. For instance, we may assume that the supplier's email account will eventually be compromised (*IsCompromised(supplier, emailAccount)*) or that their terminal will be hacked (*HasRootkit(supplier, cellphone)*). Asserting a vulnerability assumption indicates that an attacker is motivated enough to bear the cost of the corresponding attack given the business context and the anticipated benefit.

Step 4: Integrate Patterns. A number of *security workflow patterns* embodying best practices for fulfilling information exchange requirements are assumed to be available for adoption and use. Each such pattern is identified by a generic functional goal, such as *Transmit Document [Sender, Recipient, Document]*, which is decomposed into an AND/OR goal tree that describes alternative ways to fulfill the goal under various levels of security (from insecure to very secure). During matching, analysts simply connect their domain specific information exchange goals, such as *Send Invoice to Contractor* into an appropriately instantiated pattern root – in our case, *Transmit Document [supplier, contractor, invoice].* In addition, a set of expert/community maintained *attack trees* [7] are available for adoption. These are goal decompositions describing attack strategies against the workflow objective (*Transmit Document*) with respect to the different security services, such as *Confidentiality* or *Integrity*, we saw above. Based on what services are relevant for the information exchange task at hand (Step 2), the appropriate attack tree is picked and incorporated, e.g., *Compromise Integrity [Sender, Recipient, Document]* for *Integrity.*

Steps 5 and 6: Compile and Reason. The information gathered in the above steps is compiled into an AI planning specification that allows automated extraction of a workflow that (a) fulfills the information exchange requirement and, (b) picks a workflow pattern instantiation that makes the attack tree corresponding to the required security service under the specified vulnerability and domain assumptions infeasible. We utilize a Hierarchical Task Network (HTN) Planner, SHOP2 [8]. Compilation is based on a set of automatable formal translation rules. The output of the planner is a sequence of actions corresponding to the leaf level tasks of a solution to the workflow pattern decomposition tree that the planner deems suitable under the given security requirements and vulnerability assumptions.

Examples. In our invoice exchange example, assume that we want protection against unauthorized altering of the document – e.g., unauthorized change of the bank account to which payment must be made. The corresponding security service requirement, *Integrity*, is specified in the part of the STS-ml model that describes this exchange (Step 2). Subsequently, the workflow pattern *Transmit Document [supplier, contractor, invoice]* and the associated attack tree concerned with integrity *Compromise Integrity [supplier, contractor, document]* are adopted (Step 4). The reasoner (Steps 5 and then 6) will produce workflows in the form of recommended task sequences. The level of security in these workflows, however, depends on the vulnerability and domain assumptions (Step 3). If we make no vulnerability assumptions, a simple plan that involves emailing the invoice in plaintext may be recommended – the most usable option. However, we may assume that there are actors motivated and able to perform the attack. One attacker approach is to compromise the sender's and/or recipient's email accounts to allow tampering to happen. We hence assume that email accounts are vulnerable. In that case, the reasoner will sacrifice usability and recommend participants to digitally sign and verify the invoice after also engaging in key generation and exchange tasks. If, on the other hand, the *Confidentiality* service was deemed needed at the STS-ml level, plans will include encryption and decryption of the message. Importantly, the type of authentication and encryption techniques (symmetric or asymmetric) may depend on the presence of a secure channel for exchanging a secret key or practical aspects such as the software available at each side, represented through vulnerability and domain assumptions, respectively. Examples showing more framework details and how exactly the reasoning output looks like can be found in the associated repository [9].

3. Solution maturity and vision

The proposed framework aims at systematizing and automating generation of secure business workflow designs that are compliant to given security requirements, sound from a security perspective, and minimal vis-à-vis the security goals and threats – i.e., they do not include usability-compromising security steps that are unnecessary. Our approach extends existing methods for goal-oriented modeling and reasoning with security requirements, e.g., [4, 10], by including implementation concerns within its scope. A similar approach, but with a focus on architectural design, has been proposed within the UMLSec context [11]. The use of patterns has otherwise been restricted to requirements or high-level design [3, 12] rather than workflow detail. On the other hand, the rich literature on formal verification of security protocols (e.g., [13]) typically leaves requirements and context outside its scope. Finally, AI planners have been used for generating attack plans [14], rather than security designs as we do here.

Our so far work has focused on employing basic cryptographic primitives for information exchange, where we have been able to generate proof-of-concept patterns and reasoning scenarios – see [9]. However, the same approach can potentially be applied to a variety of security concerns, with a wider range of security- and privacy-enabling primitives and actions, and for different use scenarios such as software design or business process design.

There are some key challenges in realizing this vision. One is identifying the right level of model expressiveness while keeping automated reasoning tractable. The latter can be assisted by stronger search control offered by both HTNs and alternative reasoning systems through, e.g., explicating exemplar workflows versus leaving the reasoner to construct such. A second challenge is to establish a range of patterns and generic attack trees that are valid with respect to the security techniques they represent. This step requires consultation with experts and the literature, followed by testing of the reasoning outputs for correctness. Finally, critical to the empirical evaluation and application of the framework is the development of tools both for usable model development and synthesis and for fully automating the translation component.

References

[1] F. Dalpiaz, E. Paja, P. Giorgini, Security Requirements Engineering: Designing Secure Socio-Technical Systems, MIT Press, 2016.

[2] H. Mouratidis, P. Giorgini, Secure Tropos: A Security-Oriented Extension of the Tropos Methodology, Int. Journal of Software Engineering and Knowledge Eng. 17 (2007) 285–309.

[3] T. Li, E. Paja, J. Mylopoulos, J. Horkoff, K. Beckers, Security Attack Analysis using Attack Patterns, in: Proceedings of the 10th IEEE International Conference on Research Challenges in Information Science (RCIS'16), Grenoble, France, 2016, pp. 1–13.

[4] F. Massacci, J. Mylopoulos, N. Zannone, Security requirements engineering: The SI* modeling language and the Secure Tropos methodology, Studies in Computational Intelligence 265 (2010) 147–174. doi:10.1007/978-3-642-05183-8_6.

[5] E. Paja, F. Dalpiaz, P. Giorgini, Modelling and reasoning about security requirements in socio-technical systems, Data and Knowledge Engineering 98 (2015) 123–143.

[6] F. Dalpiaz, X. Franch, J. Horkoff, iStar 2.0 Language Guide, The Computing Research Repository (CoRR) abs/1605.0 (2016). URL: http://arxiv.org/abs/1605.07767. arXiv:1605.07767.

[7] B. Schneier, Attack Trees: Modelling Security Threats, 1999. URL: https://www.schneier.com/academic/archives/1999/12/attack_trees.html.

[8] D. S. Nau, T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, F. Yaman, SHOP2: An HTN Planning System, Journal of Artificial Intelligence Research (JAIR) 20 (2003) 379–404.

[9] S. Liaskos, SecureFuse: model-driven generation of secure workflow designs using security implementation patterns - Code Repo, https://github.com/cmg-york/secure-fuse/, 2022.

[10] V. Bryl, F. Massacci, J. Mylopoulos, N. Zannone, Designing security requirements models through planning, in: Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06), Luxembourg, Luxembourg, 2006.

[11] H. Schmidt, J. Jürjens, Connecting Security Requirements Analysis and Secure Design using Patterns and UMLsec, in: Proceedings of the 23rd International Conference on Advanced Information Systems Engineering (CAiSE'11), London, UK, 2011, pp. 367–382.

[12] M. Weiss, H. Mouratidis, Selecting security patterns that fulfill security requirements, in: Proceedings of the 16th IEEE International Requirements Engineering Conference (RE'08), Barcelona, Spain, 2008, pp. 169–172. doi:10.1109/RE.2008.32.

[13] C. Meadows, Formal methods for cryptographic protocol analysis: Emerging issues and trends, IEEE Journal on Selected Areas in Communications 21 (2003) 44–54.

[14] M. S. Boddy, J. Gohde, T. Haigh, S. A. Harp, Course of action generation for cyber security using classical planning, in: In Proc. of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005), AAAI, Monterey, California, USA, 2005, pp. 12–21.