Automated Generation of Attack Routes for Service Security Analysis - A Preliminary Report

Tong Li[1], Golnaz Elahi[2], Lin Liu[1], Eric Yu[3],

[1] School of Software, Tsinghua University, Beijing, China, 100084
{litong08, linliu}@tsinghua.edu.cn
[2] Department of Computer Science, University of Toronto, Canada
gelahi@cs.toronto.edu
[3] Faculty of Information, University of Toronto, Canada
yu@ischool.utoronto.ca

Abstract. i* modeling has been used to characterize service-oriented computing in terms of intentional concepts such as agents, goals, dependencies, as well as services they provide or consume. The intentional models provide a rich basis for various security related reasoning, such as vulnerability analysis, attack and countermeasure evaluation, risk assessment, etc. In this work, we aim to explore a reasoning method over the i* models that goes beyond evaluating the satisfaction of security properties. We propose a service security modeling approach for automated generation of attack routes against a specific service. We analyze the security level for each service by using the resulting models. We aim to discover countermeasures and incorporate them into the security analysis process.

1 Introduction

The i* framework offers a tool box for modeling social, organizational, or software system agents, their intentions, actions, and dependencies to other actors to achieve their goals. The i* framework does not only provide a modeling notation, but also offers a framework for thinking about systems and services: their goals, network of dependencies, alternative strategies to satisfy the goals, etc. In addition, the resulting models provide a rich basis for various types of reasoning.

The same i* modeling elements can be reused for modeling different conceptual entities such as business goals and processes [1], software services [2], software requirements [3], organizational or social relationships, knowledge entities [4], the law and regulatory [5], security goals [6,7] security vulnerabilities [6], and security attacks [6, 8, 9].

Consequently, the resulting models can be used in numerous different analysis and reasoning approaches. For example, the i* security-related models have been shown (or argued) to be useful for vulnerability analysis in social and organization networks [6], security risk assessment [9], attack analysis [6, 8], security trade-off decision analysis [7], trust analysis in the chain of dependencies [10], etc.

In most of the existing work, a model of the system is developed and certain properties under specific assumptions are checked. Several methods rely on goal

model evaluation techniques, which propagate satisfaction labels through the goal graph to check the ultimate satisfaction status of the goals. We believe the reasoning power over the i* models is not yet comprehensively explored. The i* models (with light-weight security extensions) could be useful for other types of reasoning, beyond checking some security properties. For example, the models supplied with enough security knowledge could be used for discovering security vulnerabilities, attacks, attack routes, critical entities, and countermeasures.

In this work, we aim to take the reasoning over the i* models one step ahead, and discover possible attack routes against a given service in the context of Service-Oriented Architecture (SOA). Before adopting a security mechanism, there are many questions to be answered: what are possible ways that a service can be attacked and whether they can be avoided? Are the risks high enough to adopt defensive techniques? A careful analysis of security threats at the early requirements analysis phases, before design solution is decided, would prevent adoption of (unnecessary) security mechanisms in an ad hoc way.

In this work, we propose a service security modeling framework (SSMF) as well as a reasoning method over the i* model to automatically identify the potential attack routes to a particular service. An attack route includes a path of task decompositions (AND/OR) and delegations to satisfy a top anti-service. In this method, the attacker is treated as an agent in the service environment who can conduct reasoning to meet his goals by using his capabilities. For example, the attacker composes several distributed attack actions to meet his higher level attack goal. Ultimately, based on the resulting attack routes, potential countermeasures are identified. The service compositions are then assessed with the presence of discovered countermeasures. This iterative process of identifying attack routes and countermeasures continues until the risk level of the potential attacks is tolerable. This paper reports on the ongoing work toward the discussed service security modeling framework (SSMF).

2 Research Objective

This work aims to analyze the system from the attackers' point of view and discover possible attack routes in a given service-oriented system. We model services using the i* notation and express capabilities and requirements of each service provider and consumer. Then, a hypothetical attacker is added to the service environment. The attacker has malicious goals which threaten specific services in the service environment. We use rule-based reasoning to assess whether the attacker can achieve his malicious goals. The results of this reasoning would help us to decide if the risks are high and whether security countermeasures are needed to prevent the attacks.

The main objectives of this work are toward two main directions: our first goal is to help security analysts to identify potential attack routes against a particular service. Unlike the traditional risk propagation analysis that focuses on the effects of a specific given risk, our method focuses on identifying the risks that threaten a specific given service. To discover the attack routes, we treat the attacker as an agent in the service environment who can conduct reasoning to meet his requirements using his own capabilities or by delegating some services to other agents.

The other main objective of this work is to identify the required countermeasures to defend the specific services from the attacks. The resulting attack routes are checked to see whether the top malicious goals of the attacker are satisfied. Then a number of counter attacks are added to service models of the actors which are under the attacks. In another round of evaluation, the security goals of actors under the attacks are assessed with the presence of discovered countermeasures. This iterative process of identifying attack routes and countermeasures continues until the risk level of potential attacks is tolerable.

3 The Contribution: Service Security Modeling Framework

In order to analyze the security problems in a service-oriented system and conduct automatic reasoning, we need to model and formalize the service environment. Based on the service requirements modeling ontology (SRMO) [11], we introduce some (security and service related) concepts and adjust basic concepts to aid security analysis, and build a new framework SSMF.

3.1 The SSMF Concepts

The SSMF formally defines the required concepts for analyzing threats from the point of view of the attacker. An *actor* denotes the one who carries out actions to fulfill its requirements with its capabilities. *Services* denote tasks or goals which can be required or provided in service environment. So services can be refined using *AND/OR decompositions*. An actor may have some *capabilities* to provide some services, or *Require* some services. In the latter, the actor can *delegate* his required services to other actors. For security-specific analysis, the concept of *malicious actor*, as a type of actor is considered. Malicious Actors focus on attacking other actors' services. An attack is defined as a triple relation, which involves an actor, a *malicious task* and a service (under the attack). Malicious tasks (or a non-malicious service) may *obstruct* other services.

Actors may have knowledge about certain facts, e.g., services, decomposition of a service, who can provide a service, etc. The knowledge assumptions are later used for attack reasoning and generating potential attack routes. To enable the reasoning and generation of attack routes, we have defined three operations: *add* which inserts a new piece of knowledge into the knowledge set of an actor; *conduct* which decomposes a service; and *satisfy* which is used to represent an actor has satisfied has required service.

3.2 Service Security Analysis

The concepts and operations introduced earlier would provide the bed to reason about the possible attacks in a service-oriented setting. We have organized the service security analysis into four steps: 1) Scenario and environment modeling; 2) Attack goal identification; 3) Attack reasoning from the attacker's point of view; 4) Attack identification and assessment. Fig. 1 illustrates the discussed process for a search service in the web environment.

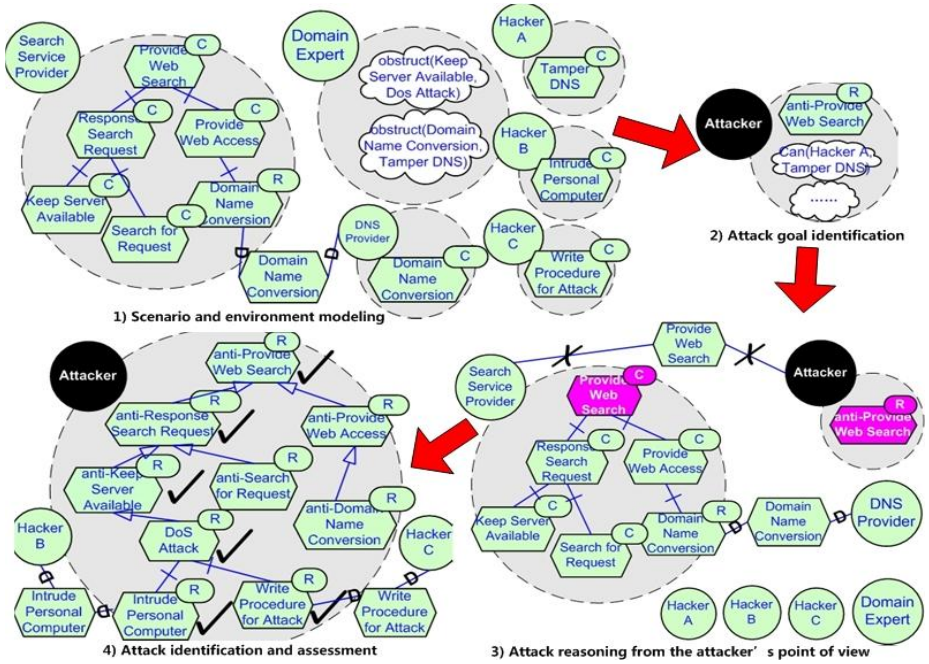

Fig. 1. Service security modeling and reasoning for a web search service example.

1) Scenario and environment modeling. The first step in service security analysis is to model the service environment, i.e. modeling the capabilities and required services within the actors' boundaries, modeling the interactions (delegation of services) among the actors, and formalizing the knowledge each actor hold.

2) Attack goal identification. In the next step, a number of services for which we need to analyze attacks and the security level are selected. Then, the anti-services against the target services are generated. These anti-services are added to a hypothetical attacker which requires achieving the anti-services.

3) Attack reasoning from the attacker's point of view. Since attackers are external entities and we do not have sure knowledge about their capabilities and level of knowledge, we consider the worst possible case in which attackers have enough knowledge in the service environment. In order to discover the attack routes and assess whether an attacker is able to satisfy his ultimate anti-service requirements, we define a number of reasoning rules. The predicates of these production rules are either the specification of services' refinement or available knowledge about possible attacks. The reasoning rules are used to refine the anti-service into malicious tasks and services, delegate the required malicious services of attackers to other actors, and add a piece of knowledge to the attackers' knowledge set. Finally, the satisfaction propagation rule is used to check if the discovered attack route would ultimately satisfy the top anti-services.

4) Attack identification and assessment. Given the discovered attack routes, we can assess the severity or probability of each possible attack route, and accordingly, decide on proper countermeasures. If the attack reasoning cannot uncover a feasible attack route, it proves that the target service is safe enough in the service environment.

4 Conclusions, Limitation, and Ongoing and Future Work

This paper reports on an ongoing work to develop a reasoning method over i* models analyzing the risks against a given services in a SOA setting. In this work, i* models supplied with enough knowledge about the attacks are used for automatic generation of possible attack routes. The results of the reasoning would enlighten the required countermeasures to protect the services under the attack.

However, the current reasoning method and the set of rules we have developed do not support automatic discovery of the countermeasures. Besides, iterative analysis of countermeasures and re-generation and assessment of possible attacks with the presence of countermeasures is not yet incorporated into the method. The other shortcoming of this work is focusing only on the attacks that obstruct a service and make it unavailable. In real world, an attack may threaten integrity or confidentiality of the data that is exchanged or produced by a service. Our next step is to formalize the impacts of malicious tasks against integrity and confidentiality of services, and define the required rules for automatic generation of attacks that threaten these two properties. Finally, we aim to implement the proposed reasoning method using the JESS reasoning engine which uses Rete algorithm to process rules and written in Java.

References

1. Grau, G., X. Franch, and N.A.M. Maiden, *PRiM: An i*-based process reengineering method for information systems specification.* Information and Software Technology, 2008. 50(1-2): p. 76-100.
2. Wang, P., et al., *Building toward Capability Specifications of Web Services Based on an Environment Ontology.* IEEE Trans. on Knowl. and Data Eng., 2008. 20(4): p. 547-561.
3. Yu, E.S.K., *Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering*, in *Proc. of the 3rd IEEE Int. Symposium on Requirements Engineering*. 1997, IEEE Computer Society. p. 226.
4. Strohmaier, M., et al., *Analyzing Knowledge Transfer Effectiveness--An Agent-Oriented Modeling Approach*, in *Proc. of the 40th Annual Hawaii International Conference on System Sciences*. 2007, IEEE Computer Society. p. 188b.
5. Ghanavati, S., D. Amyot, and L. Peyton, *Compliance Analysis Based on a Goal-oriented Requirement Language Evaluation Methodology*, in *Proc. of Int. RE Conf.,* IEEE Computer Society. p. 133-142.
6. Liu, L., E. Yu, and J. Mylopoulos. *Security and privacy requirements analysis within a social setting*, in *Proc. of Int. RE Conf.*, 2003.
7. Elahi, G. and E. Yu, *Modeling and analysis of security trade-offs - A goal oriented approach.* Data Knowl. Eng., 2009. 68(7): p. 579-598.
8. Elahi, G., E. Yu, and N. Zannone, *A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities.* REJ, 2010. 15(1): p. 41-62.
9. Matulevičius, R., et al., *Adapting Secure Tropos for Security Risk Management in the Early Phases of Information Systems Development*. 2008. p. 541-555.
10. Giorgini, P., et al., *Modeling Security Requirements Through Ownership, Permission and Delegation*, in *Proc. of Int. Conf. on RE*. 2005, IEEE Computer Society. p. 167-176.
11. Liu, L., et al., *Towards a service requirements modelling ontology based on agent knowledge and intentions.* International Journal of Agent-Oriented Software Engineering, 2008. 2(3): p. 324-349.