A Holistic Approach to Security Attack Modeling and Analysis *

Tong Li[1], Jennifer Horkoff[2], Kristian Beckers[3], Elda Paja[1], and John Mylopoulos[1]

[1] University of Trento, Trento, Italy
{tong.li,paja,jm}@unitn.it
[2] City University London, London, UK
horkoff@city.ac.uk
[3] Technische Universität München, München, Germany
beckersk@in.tum.de

Abstract. Protecting socio-technical systems is a challenging task, as a single vulnerability or exposure of any component of the systems can lead to serious security breaches. This problem is exacerbated by the fact that the system development community has not kept up with advances in attack tactics. In this paper, we present ongoing research on the development of a holistic attack analysis technique. Our approach adopts a goal modeling technique to capture attacker malicious intention as anti-goals, which are systematically refined and operationalized into concrete attack actions which target various assets (e.g., human, software, and hardware). A comprehensive attack pattern repository (CAPEC) is seamlessly integrated into our approach in order to provide analysts with practical security knowledge and assist them in identifying potential attacks under specific contexts. Finally, a set of security controls is provided for mitigating identified attacks.

1 Introduction

Socio-Technical Systems (STSs) consist of human, software and physical elements that together fulfill system requirements. Due to their heterogeneity and complexity, such systems are exposed to a broader range of attacks than their software cousins. Attackers are able to breach system security by targeting any vulnerable component of STSs, such as human, software applications, or physical infrastructure. Consider a smart meter system as an example [1]. An attacker can access energy consumption data by performing social engineering against the stakeholders, by intercepting communication data transmitted between software applications, or even by probing the physical smart meter device. The larger attack surfaces of STSs can also lead to multistage attacks that combine attacks on different parts of an STS [2].

*Copyright © 2015 for this paper by its authors. Copying permitted for private and academic purposes.

Thinking like an attacker has been proposed as an effective solution to discover attacks that are most likely to be performed by an attacker [3]. As such, security analysis can consider alternative countermeasures to mitigate identified attacks and ensure the satisfaction of security requirements. Many approaches have been proposed for analyzing security requirements from an attacker's perspective, such as anti-goal analysis [4] and misuse cases [5]. However, these approaches are not designed for STSs but for software, i.e., do not explicitly capture inter-dependencies between software and other system components (e.g., business processes, hardware). As a result, multistage attacks that target several system components cannot be appropriately captured.

Another obstacle to STS security is that attack analysis lacks knowledge of impending attacks. Barnum and Sethi have pointed out that the software engineering community has not kept up with advances in attack knowledge, resulting in less effective, and sometimes useless security designs [3]. Attack patterns, as solutions to this problem, document reusable attack knowledge in support of system security solutions. Specifically, CAPEC (Common Attack Pattern Enumeration and Classification) is a comprehensive attack knowledge repository, which includes 463 attack patterns[1]. However, without an efficient method to use this large set of patterns, analysts are reluctant to adopt them in practice [6].

We have proposed a holistic approach for modeling and analyzing attacks for STSs in a companion poster [7]. In this paper, we describe recent progress regarding this work. In particular, we present and illustrate a refined analysis process. We base our approach on a three-layer requirements framework [8] in order to consider threats from various system viewpoints and provide a holistic security analysis. Specifically, our approach takes an attacker's viewpoint to generate attack strategies by systematically capturing and refining attacker malicious intentions. Moreover, we seamlessly integrate CAPEC attack patterns into our approach to effectively identify operational attacks, based on which corresponding security controls are applied. Finally, a supporting tool is under development, and we also describe how the tool can support the proposed analysis process.

2 Background

Three-layer requirements modeling framework. Li et al. [8] proposed a three-layer requirements framework, which models and analyzes requirements of STSs at the business layer, software application layer, and physical infrastructure layer, respectively. In our proposal, we take the three-layer requirements model as input, which allows us to capture threats that originate in different layers of a system, and analyze attacks from a holistic viewpoint.

Contextual goal modeling. Ali et al. [9] have extended Tropos with context-related concepts in order to model and analyze stakeholder requirements in different contexts. In this paper, we propose to model attack patterns as contextual

[1]https://capec.mitre.org

goal models in order to (semi-)automate the selection of attack patterns during anti-goal analysis and operationalization.

Attack patterns. Inspired by design patterns, attack patterns were first proposed by Moore et al. [10] in order to reuse proven attack knowledge. Notably, CAPEC has been under development for years and currently includes 463 attack patterns. Each attack pattern is specified in terms of *Attack Prerequisites*, *Attack Motivation-Consequences*, *Solutions and Mitigations* etc. However, it is difficult to use the CAPEC repository, as analysts have to manually navigate and select appropriate patterns. In this paper, we model attack patterns as contextual goal models in order to semi-automate the corresponding analysis.

3 A Holistic Attack Analysis Approach

Our approach takes a three-layer system requirements model as input, which includes both functional requirements and security requirements, and eventually produces a list of security controls that can effectively protect the system from being damaged by attackers. An overview of the holistic attack analysis process is shown in Fig. 1. Each step of the analysis will be specified in the following subsections.

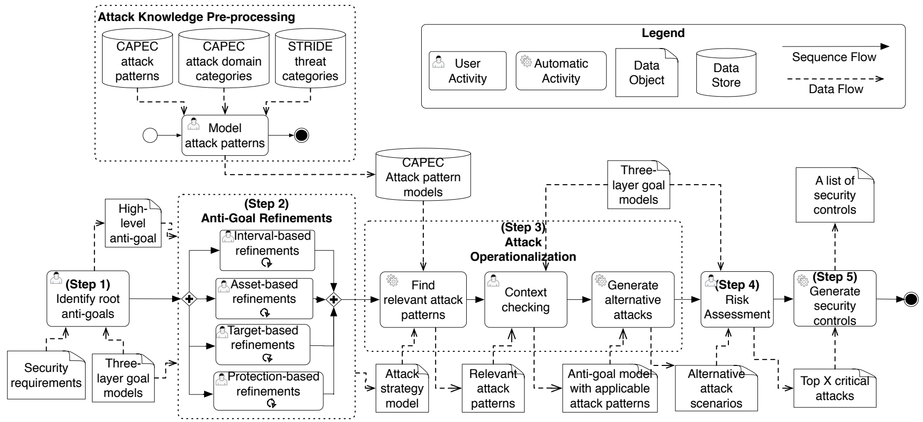
Attack Knowledge Pre-processing
CAPEC attack patterns
CAPEC attack domain categories
STRIDE threat categories
Model attack patterns
Legend
User Activity
Automatic Activity
Data Object
Data Store
Sequence Flow
Data Flow
CAPEC Attack pattern models
Three-layer goal models
A list of security controls
High-level anti-goal
(Step 2) Anti-Goal Refinements
Interval-based refinements
Asset-based refinements
Target-based refinements
Protection-based refinements
(Step 1) Identify root anti-goals
(Step 3) Attack Operationalization
Find relevant attack patterns
Context checking
Generate alternative attacks
(Step 4) Risk Assessment
(Step 5) Generate security controls
Security requirements
Three-layer goal models
Attack strategy model
Relevant attack patterns
Anti-goal model with applicable attack patterns
Alternative attack scenarios
Top X critical attacks

Fig. 1: An overview of the holistic analysis process

Step 1: Root anti-goal identification. We capture an attacker's high-level malicious intentions against a system as anti-goals, which are systematically analyzed to explore alternative attacks. In order to (semi-)automate anti-goal analysis, we propose to characterize attacker's anti-goals as a quadruple, consisting of four attributes: *Asset*, *Threat*, *Target*, and *Interval*.

– *Asset* is anything of value to stakeholders. Attackers can benefit from attacking assets.
– *Threat* indicates an undesired condition of an asset, which attackers try to achieve to fulfill their malicious desires. In this work, we leverage the STRIDE threat categories [6] to specify threats.

- *Target* is a component of a system, which involves assets and has vulnerabilities that are exploitable by attackers. Within the three-layer system structure, targets vary from layer to layer.
- *Interval* represents the time period, during which attackers carry out attacks. In this work, an interval is specified in terms of a system functional task, which indicates the execution period of the task. Note that a goal can also be specified as an interval, which means the execution period of all the operationalized tasks of this goal.

As shown in Fig. 2, a root anti-goal *AG1* is derived from the root security goal *SG1* that is captured in the three-layer goal model. In particular, *AG1* capture the malicious intention "Tampering (*Threat*) energy demand (*Asset*) when the real-time pricing is applied (*Interval*) by attacking the energy supplier (*Target*)", which negates *SG1*.

Step 2: Anti-goal refinement. When attacking a complex system, an attacker can have various attack strategies to achieve his root anti-goal. An attack strategy sheds light on which system components to attack and when to attack, but does not mention concrete techniques and attack actions. Once root anti-goals are identified, we propose to systematically refine them in order to explore various attack strategies across three layers.

To this end, we investigate several attack scenarios (reported in [2]) to understand how attackers generate attack strategies to achieve their malicious intention. Based on the investigation, we identify four refinement methods to simulate the generation of attack strategies, as presented in Fig. 1. Take the interval-based refinement pattern as an example. As shown in Fig. 2, the root anti-goal *AG1* applies during the interval *G1*, and the interval *G1* is "and-refined" into two sub-interval *G2* and *G3*. Thus, *AG1* is "or-refined" into *AG2* and *AG3*, which apply during intervals *interval(G2)*, *interval(G3)* respectively.

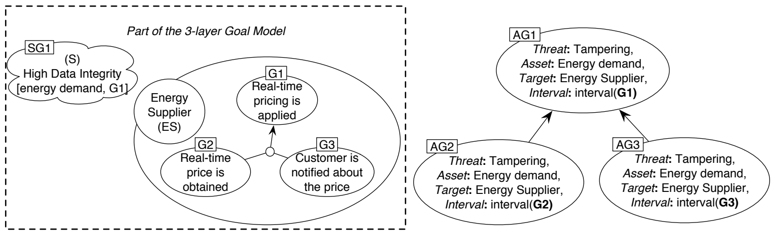

Fig. 2: An example of interval-based refinement

Step 3: Anti-goal operationalization. Anti-goal refinements address when and what to attack in order to achieve attacker malicious intentions. In this step, we leverage the attack knowledge from the CAPEC repository to analyze whether leaf anti-goals can be achieved by known attacks. In order to (semi-) automate the analysis, we construct a contextual goal model for each CAPEC attack pattern according to its textual description. An example is shown in Fig. 3, capturing the JSON Hijacking pattern.

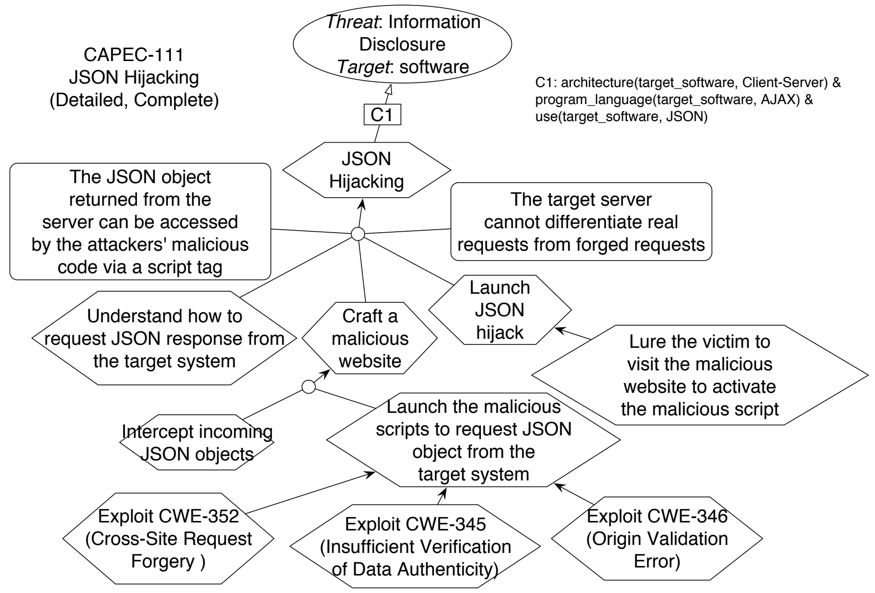
CAPEC-111
JSON Hijacking
(Detailed, Complete)
Threat: Information Disclosure
Target: software
C1
C1: architecture(target_software, Client-Server) & program_language(target_software, AJAX) & use(target_software, JSON)
JSON Hijacking
The JSON object returned from the server can be accessed by the attackers' malicious code via a script tag
The target server cannot differentiate real requests from forged requests
Understand how to request JSON response from the target system
Craft a malicious website
Launch JSON hijack
Lure the victim to visit the malicious website to activate the malicious script
Intercept incoming JSON objects
Launch the malicious scripts to request JSON object from the target system
Exploit CWE-352 (Cross-Site Request Forgery)
Exploit CWE-345 (Insufficient Verification of Data Authenticity)
Exploit CWE-346 (Origin Validation Error)

Fig. 3: The attack pattern model of JSON Hijacking (CAPEC-111)

Operationalization analysis consists of two steps: relevance analysis and applicability analysis. Given a leaf anti-goal, we first automatically identify all relevant attack patterns from the attack repository. In particular, if the leaf anti-goal and the anti-goals modeled in an attack pattern model concern the same threat and the same type of target (software, hardware etc.), then the that attack pattern is relevant to the leaf anti-goal. After identifying the relevant attack patterns, we further check their applicability, i.e., whether the contexts required by the attack patterns are held in the target system. Specifically, the context of each relevant pattern will be automatically checked against the three-layer requirements goal model. If the information captured in the goal model is not enough to determine whether the context applies, then the supporting tool will interactively ask analysts to check them. Once the applicable attack patterns are determined, we can automatically generate all alternative attacks according to the derived anti-goal model.

Step 3: Risk assessment. For alternative attacks identified in the previous step, we assess their risk by analyzing the vulnerabilities exploited by those attacks. To this end, we propose to use external vulnerability assessment services to detect vulnerabilities that exist in the target system [2]. Based on the result of vulnerability analysis, we can assess the risk of alternative attacks and further prioritize them.

Step 4: Generate mitigation controls. Given prioritized alternative attacks, an analyst can choose how many to tackle, depending on her budget. For each

[2]https://cve.mitre.org/compatible/product_type.html

attack to be addressed, our approach finally generates corresponding mitigation controls according to security knowledge documented in the CAPEC repository.

4 Conclusions and Future Work

We present ongoing research on a holistic attack analysis technique, which takes an attacker's viewpoint by capturing their malicious intents as anti-goals. The approach takes into account threats that originate from various system components, identifies alternative attacks that target the vulnerable components, and finally provides effective security controls to satisfy security requirements.

Apart from the analysis process we have investigated, we are working on the tool-supported implementation of each step. In particular, we seek to deeply integrate practical attack knowledge (e.g., CAPEC) into our approach in order to deal with real world security problems. To this end, we need to process a reasonable amount of the attack patterns (as presented in Section 3). Furthermore, we will improve the integration of external vulnerability assessment services into the risk assessment step of our analysis process. Once all the analysis steps are well designed and can be supported by our tool, we plan to perform a case study to validate our approach.

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

References

1. Flick, T., Morehouse, J.: Securing the smart grid: next generation power grid security. Elsevier (2010)
2. Mitnick, K.D., Simon, W.L.: The art of deception: Controlling the human element of security. John Wiley & Sons (2011)
3. Barnum, S., Sethi, A.: Attack patterns as a knowledge resource for building secure software. In: OMG Software Assurance Workshop: Cigital. (2007)
4. Lamsweerde, A.V.: Elaborating security requirements by construction of intentional anti-models. In: ICSE. (2004) 148–157
5. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requirements Engineering **10**(1) (2005) 34–44
6. Shostack, A.: Threat Modeling: Designing for Security. John Wiley & Sons (2014)
7. Li, T., Paja, E., Mylopoulos, J., Horkoff, J., Beckers, K.: Holistic security requirements analysis: An attacker?s perspective. In: Requirements Engineering Conference (RE), 2015 IEEE 23nd International, (to be published) (2015)
8. Li, T., Horkoff, J.: Dealing with security requirements for socio-technical systems: A holistic approach. In: CAiSE'14. (2014)
9. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering **15**(4) (2010) 439–458
10. Moore, A.P., Ellison, R.J., Linger, R.C.: Attack modeling for information security and survivability. Technical report, CMU-SEI-2001-TN-001. (2001)