Towards a Semantic Web-enabled Knowledge Base to Elicit Security Requirements for Misuse Cases

Haibo Hu\textsuperscript{1}, Dan Yang\textsuperscript{1}, Hong Xiang\textsuperscript{1}, Li Fu\textsuperscript{1}, Chunxiao Ye\textsuperscript{2} and Ren Li\textsuperscript{2}

\textsuperscript{1} The School of Software Engineering, Huxi Campus of Chongqing University
410331, Chongqing, China

\textsuperscript{2} College of Computer Science, Chongqing University
400030, Chongqing, China

Abstract. Eliciting security requirements is critical but hard for non-expert to fulfill an exhaustive analysis on large body of security knowledge. Emerging models in requirements engineering (Re) society release some burden of such difficulty, as well as security ontologies are booming for knowledge sharing and reuse. There exists necessity for the synergy of them, such as utilizing security ontology (So) as the back end of Knowledge Base (KB) for capturing security requirements by using known Re models. Research advances in the Semantic Web (Sw) community provide a common framework of technologies that allows data to be shared and reused across boundaries of various application and community. This paper proposes a knowledge base which is constructed on So and Misuse Case Model (MCM), by representing them into OWL (Web Ontology Language). Semantic rules can be derived from the correlation of So and MCM to be utilized for reasoning and querying security knowledge via MCM-based requirements elicitation. The proposed KB coordinates So with a specific Re model to facilitate knowledge sharing to be a foundation for eliciting security requirements automatically.

1 Introduction

In the last decades, researchers and practitioners have been aware that security concerns in software system must be taken into account at the very beginning of development life cycle, due to adding security features in ad-hoc manner at later stages is costive in term of time and resources [1][2]. Most requirements engineers and software developers are not primarily interested in security domain [3], or poorly trained to elicit, analyze, and specify security requirements, often confusing them with the architectural security mechanisms [4]. Thus it’s hard for a non-expert to understand jargons of security issues, such as “under which circumstances does an information asset have vulnerability that can be utilized by attackers to form threats, and how to find adaptable mechanisms to defend or to retrieve a policy to mitigate the risk?”

The state-of-the-art works show research trend in two directions to overcome the bottle neck. In their systematic literature review, Fabian et al [5] and Mellado et al [6]...
survey one direction which is mainly derived from RE community that extends traditional methodologies of requirements analysis to manifest security concerns. These work includes adopting anti-models to goal-oriented approaches [7], introducing Misuse Case [8] to existing Use Case models (UCM), customizing UML profiles as to UMLsec [10], reinforcing Goal/Agent-oriented model like i*/Tropos with the capability of security analysis [11-14], or using Abuse Frame [15] to extend Problem Frame, etc. The other direction is mainly located in security engineering domain which aims to represent and manage security knowledge for reuse. This direction leads to several approaches such as developing security patterns [16] for abstraction or reuse, engineering SOs for knowledge sharing [17], and representing security knowledge with the SW technologies for reasoning and querying [18].

While engineering security requirements by means of known RE models (such as KAOS [19], i*/[20] Tropos [21], Use Case [22], or Problem Frame [23], etc.) stress their objectives on modeling the system-to-be requirements by taking security as constrains, thus lacking of facilities represents complex cause-and-effect relationships for security issues. Moreover, it’s costive to train stakeholders with adequate security knowledge to fulfill security analysis in requirement stage.

We aim to facilitate engineering security requirements with known RE models as MCM by means of security ontology for knowledge management. Security requirements to be modeled into misuse cases can be elicited in an automatic way by mapping existing use cases into a specialized ontological security KB for reasoning and querying. We map core concepts of MCM to the SO developed by Herzog et al [17] and Lasheras et al [18], to rebuild security core ontology for the T BOX [25] of the KB. The KB is qualified with semantic rule-based languages (SWRL and SQWRL) for reasoning and querying. In real usage of the KB, we put refined use cases as instances in the A BOX [25], and execute predefined rules to reason and query knowledge for misuse cases. The framework of our proposed method is illustrated in Fig. 1.

Fig. 1. Framework of the proposed methods.

The advantages of our proposed approach are twofold: on the one hand this method offers a step-by-step process for modeling ontologies and requirements in U/MCM as well as bridging concepts among them onto semantic level. Based on this preliminary
process, the framework also shows an approach for eliciting security requirements with misuse cases into automation.

2 Related Works

In this section, we present literature review of related works both on security ontologies and misuse case for security RE, as well as research advances in the semantic web community that are utilized in our work for constructing the knowledge base.

MCM is the inverse or anti-model of UCM [22] which is introduced by Sindre and Opdahl [8] in late 1990’s. The required behaviour of software under development specified with UCM which is essentially structured stories or scenarios describing what should happen when the software or product is used. On the contrary, a misuse case describes a negative scenario that should not happen, and identify system threats thus leads to new requirements expressed in use cases for mitigation. It’s a good tool to treat non-functional requirements [26] from the very beginning of development life cycle by avoiding premature design decisions [27]. The significance of employing MCM is that it enhances the communication between the developers and the stakeholders to agree on critical system solutions by regarding the trade-off analysis [28], and relates well with UCM and UML for Model Driven Development of secure software system [29-30]. Visaggio and de Rosa introduce a system to capture and reason software security knowledge for MCM [31-32], by means of similarity function.

In the last decade, Ontology as a methodology has applied to security engineering in a broad range. The benefits of employing Ontology for security domain are knowledge representation, sharing and reuse. In terms of derivation for vocabulary, security standards play important roles, such as Common Criteria (CC) [33] and BS7799 [34]. While representing relationships of these concepts owes much more to security models or framework of best practices, such as OCTAVE [35], MAGERIT [34], or CORAS [37].

Known SOs are surveyed by Blanco et al in [24], with their applications in various domains, such as access control modelling and reasoning [39], security management [40], intrusion detection [41], web services [42], and security RE [18].

To the best of our knowledge, there is not so much works on the pattern of synergistic above directions as SO2RE. Among these relate works, there are two fruits that are most relevant to our work. The first one is Lasheras et al’s SO framework [18] for reusing security requirements. However, this framework concentrates on managing security knowledge, without concern of any security requirement models. The other is Visaggio et al’s knowledge management system to capture and reason security knowledge for MCM [32], while the system is based on similarity function instead of ontological and SW-based technology.

The current set of W3C standards is based on RDF [43], a language that provides a basic capability of specifying graphs with a simple interpretation and serializing them in XML. The OWL [44] is a family of knowledge representation languages based on Description Logics (DL) [43] with a representation in RDF. OWL supports the specification and use of ontologies that consist of terms representing individuals, classes of individuals, properties, and axioms that assert constraints over them. The axioms can be realized as simple assertions or simple rules. Semantic Web Rule Language (SWRL)
is a proposal for a Semantic Web rules-language, combining sublanguages of the OWL with those of the Rule Markup Language (RuleML) [46]. Rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. SQWRL (Semantic Query-enhanced Web Rule Language) [47] is built on the SWRL rule language. SQWRL takes a standard SWRL rule antecedent and effectively treats it as a pattern specification for a query. In addition, standard SWRL serialization mechanisms can be used, so queries can be stored in OWL ontologies.

3 Knowledge Base on Security Ontology for MCM

3.1 The Core Security Ontology for U/MCM

In our work, we aim to build a security knowledge base according to existing known SOs that can be used for MCM to elicit security requirements. Thus we adopt the SOs developed by Herzog et al [29] and Lasheras et al [18] whose works are mainly based on security risk analysis model with core concepts as Asset, Threat, Vulnerability and Countermeasure. In order to cooperate with MCM, we also add some other concepts discussed in [31] as Attack, Attacker and Security Goal to the core ontology.

The core SO should represent relationships of above concepts capable of expressing fair knowledge for MCM. Fig. 2 shows the core concepts and relationships of this core SO with UML class diagram.

![Core Security Ontology used for the Knowledge Base.](image)

3.2 Representing UCM/MCM in the SW

In our work, we represent the Ontology of U/MCM in OWL to facilitate the KB with capability of reasoning and querying. In order to represent it without ambiguity, the core ontology of UCM/MCM is defined with some axioms derived from related works.
These axioms are rules in syntax level for making us choose a pattern or manner to design the Ontology, as shown in Table 1.

### Table 1. Axioms in the Core Ontology of Use/Misuse Case Model.

<table>
<thead>
<tr>
<th>Axioms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Only does User associate with Use Case, and Misuser with Misuse Case.</td>
</tr>
<tr>
<td>2</td>
<td>A use case may include a use case, or a security use case.</td>
</tr>
<tr>
<td>3</td>
<td>A misuse case only threatens a use case but a security use case.</td>
</tr>
<tr>
<td>4</td>
<td>Only does a security use case mitigate a misuse case.</td>
</tr>
<tr>
<td>5</td>
<td>One use case does not mitigate another use case, or a security use case.</td>
</tr>
</tbody>
</table>

Base on axioms 1, we find that it’s better to define the concepts like User and Misuser by represent the association relationship between User and Use Case, Misuser and Misuse Case prospectively. While for axioms 2-5, Security Use Case should not be a sub-class of Use Case. The correlation of Use Case, Misuse Case and Security Use Case can be represented as object properties among them. The core ontology of Use/Misuse Case is edited with Protégé 3.4 and visualized in Fig. 3.

Fig. 3. The Core Ontology of Use/Misuse Case represented in OWL.

We discuss how to fulfill the task by clarifications as well as examples, to make guidelines for refinement. Mining more facts from use cases to guide refinement of use cases is prerequisite work to make the KB works. Due to limited space, we only present two cases for illustration.

1. **Clarification**: An asset (critical asset) is vulnerable in a specific context or a given scenario for exploitation of security attack by a misuser (attacker), thus leads to threats for the system.

   **Example**: For an E-commerce system, a use case as “User Login” is specified like “user login the system by inputting user-id and password”. In this case, password should be identified into critical asset (as a type of data) that if be transmitted in plaintext (without declaring data transmission by secure mechanism) will be vulnerable for eavesdropping by attackers.

   **Refinement Guideline**: All assets should be recognized and valued in use cases, as well as be declared their context for indicating clues of vulnerabilities. In most cases, the vulnerability of an asset can be realized by non-declaration of security constrains.
(2) **Clarification**: If an asset (dependor) depends on other assets (dependees) which are vulnerable, then the dependor asset is vulnerable.  

**Example**: For the same E-commerce system, a use case as “User Reset Password” specifies “When a user forgets his password, s/he can claim to the system to reset she/his password.” This case may include another use case, i.e., “Reset Password by E-mail”, which specifies “When user claimed reset password, an acknowledgement of the claim with a URL for performing password reset is sent to the user by E-mail”. In these cases, password depends on another asset E-mail. If secure E-mail transmission is not declared, then the password is vulnerable when being reset.  

**Refinement Guideline**: The correlation of assets should be identified in a use case, as well as in use cases with associations as includes and extends relations.

### 3.3 ABOX of the Knowledge Base

The roles of ABOX are important to representing concrete real world cases in real usage other than concept modeling. There are two steps to fulfill this task, one is to add instances of security ontology and assert their relationships with constraining of concept model in TBOX, the second step is adding instances of UCM with refinement of use cases. A partial view of the ABOX in Fig. 4 shows the cases discussed.

![Fig. 4. A partial view of ABOX in the Knowledge Base.](image)

#### 4 Extracting Rules for Reasoning and Querying

In the KB, responsibilities of rules are twofold, i.e., reasoning and querying. Rules for reasoning are represented in SWRL in line with knowledge presentation in OWL aiming to find tacit knowledge for mapping use cases to SO, and to be executed to mine facts from use cases to ABOX of the KB, such as inference for vulnerable assets.

The guidelines for refining use cases discussed in sub-section 3.2 can be used to extract rules in and for the KB. For readability the rule is expressed as software code style in this paper. For example, a rule for “inferring an asset is vulnerable when it depends on another vulnerable asset” is shown as follow.
Example Rule-1: Infer vulnerability of Asset on Dependency
/* An asset is vulnerable when it depends on another vulnerable asset */
/* Antecedent of Rule-1 */
hasAsset(?uc_a, ?a) ^ hasAsset(?uc_b, ?b) ^
includes(?uc_a, ?uc_b) ^ dependsAsset(?a, ?b) ^
isVulnerableAsset(?b)
/* Consequent of Rule-1 */
• isVulnerableAsset(?a)
/* End of Rule-1 */

Requisite of or based on reasoning, rules for querying are represented in SQWRL and
objective to retrieve security knowledge to be used for use cases to elicit misuse cases,
such as querying possible security attacks on a given vulnerable asset and retrieving
 corresponding countermeasures for the asset for mitigations, as shown bellow.
Example Rule-2: Query security attacks on vulnerable assets and relevant countermeasures
/* Querying possible security attacks on a given vulnerable asset and retrieving corresponding countermeasures for the asset for mitigations */
/* Antecedent of Rule-2 */
Asset(?asset) ^ hasVulnerability(?asset, ?vul) ^
exploitsByAttack(?vul, ?att) ^ formsThreat(?att, ?thr) ^
hasCountermeasure(?vul, ?cout)
/* Consequent of Rule-2 */
/* End of Rule-2 */

5 User Interface of the Knowledge Base

Rules may be used for inferring explicit or implicit facts, depends on the granularity of
knowledge fragments that users are interested in. User Interface interacts with user and
the KB, and should be open for new features. Currently, Protégé 3.4 is employed as the
interface with capability of editing OWL ontology, defining rules and reasoning them with
third party rule engines like Jena or Jess. Screenshots of the tool are given below.

![Fig. 5. Screenshots for Interface of proposed Knowledge Base. Notes: (a) class hierarchy in TBOX; (b) refined and mapped use cases into ABOX by extracting relationships of instances.](image-url)
6 Conclusions

This paper presents a process of constructing a Kb represented in OWL by SO and for MCM. Rules for reasoning and querying can be derived from the correlation of SO and MCM as well as refinement of use cases. The proposed Kb coordinates SO with specific RE model by executing rules to elicit security requirements. The proposed framework is capable of customizing to be used for other known RE models such as i*/Tropos or NFR framework/patterns by properly mapping their concept models to SO.

Limitations of the proposed KB and method mainly attribute to the weakness of rule languages, such as decidability by safe logical constrains, and can not call each other without programs. Moreover, Protegé is a tool in general purpose which is not specific for RE, thus it can not work well on requirements refinement. In our future’s work, we consider to develop a specific tool to overcome the problems. Besides, exploitations on applying the KB to other RE model will also be taken into account.

Acknowledgements

The work is supported by the National S&T Major Project in China under grant no. 2009ZX07315-006, Nsr-China under grant no. 60803027. We also thank the anonymous reviewers for their coming helpful comments, and members in Protégé-owl mailing list for their help.

References


37. CORAS: A platform for risk analysis of security critical systems http://www2.nr.no/coras/, (last viewed on 15 Mar., 2011)