Security Requirements Analysis Using Knowledge in CAPEC

Haruhiko Kaiya[1], Sho Kono[2], Shinpei Ogata[2], Takao Okubo[3], Nobukazu Yoshioka[4], Hironori Washizaki[5], and Kenji Kaijiri[2]

[1] Dept. of Information Sciences, Kanagawa University, Hiratsuka 259-1293, Japan
kaiya@kanagawa-u.ac.jp
[2] Dept. of Computer Science, Shinshu University, Nagano 380-8553, Japan
[3] Institute of Information Securiry, Kanagawa 221-0835, Japan
[4] National Institue of Informatics (NII), Tokyo 101-8430, Japan
[5] Waseda University, Tokyo 169-8555, Japan

Abstract. Because all the requirements analysts are not the experts of security, providing security knowledge automatically is one of the effective means for supporting security requirements elicitation. We propose a method for eliciting security requirements on the basis of Common Attack Patterns Enumeration and Classification (CAPEC). A requirements analyst can automatically acquire the candidates of attacks against a functional requirement with the help of our method. Because technical terms are mainly used in the descriptions in CAPEC and usual phrases are used in the requirements descriptions, there are gaps between them. To bridge the gaps, our method contains a mapping between technical terms and noun phrases called term maps.

Keywords: Requirements Engineering, Requirements Elicitation, Security Requirements, Structured Knowledge.

1 Introduction

Because an information system is embedded and used in some social activity such as business or entertainment, knowledge of the social activity, i.e., domain knowledge is crucially necessary for eliciting the requirements for the system. Knowledge of information technology such as Internet and mobile devices is a kind of domain knowledge because our daily activities cannot be performed without such technologies.

Especially in security requirements elicitation, knowledge related to technologies plays an important role because actual attacks are achieved by means of the technologies and most countermeasures are implemented in the same way. We thus expect requirements analysts have enough knowledge of security and related technologies when he/she elicits security requirements. Apparently, it is unrealistic expectation because the knowledge of security and related technologies is too huge for all requirements analysts to understand it. In addition, such knowledge is always updated every day. Providing the knowledge of security and related technologies automatically is one of the helpful ways to satisfy this assumption.

In this paper, we proposed a method for eliciting security requirements on the basis of Common Attack Patterns Enumeration and Classification (CAPEC), which is

L. Iliadis, M. Papazoglou, and K. Pohl (Eds.): CAiSE 2014 Workshops, LNBIP 178, pp. 343–348, 2014.

machine-readable and up-to-date knowledge for attacks. CAPEC consists of attack patterns, and the patterns are usually increased when CAPEC is updated. In the method, candidates of attacks against a function (a use case) are automatically selected from the knowledge. Although requirements analysts have to examine whether each candidate actually threatens the function, they do not have to learn all the knowledge in advance.

Because attacks using concrete technologies such as scripting and database queries are focused in CAPEC, most descriptions in CAPEC consist of technical terms. However, a description of a function such as a use case description in a use case model is usually implementation-independent. To bridge the gaps between CAPEC descriptions and requirements descriptions, a mapping between technical terms and noun phrases called "term map" is used in our method. Although we have to manually prepare the term map, we assume we can gather and improve the term map step by step during its usages.

The rest of this paper is organized as follows. In section 2, we review related works using domain knowledge for eliciting requirements. In section 3, we present a method for eliciting security requirements. We finally summarize our current results and show the future issues.

2 Related Work

The importance of the domain knowledge is widely accepted in requirements elicitation and even in software engineering in general [10]. We can thus find a lot of requirements elicitation methods using domain knowledge. One of the significant problems of the methods is that there is no guarantee to develop or acquire the domain knowledge. There are a few methods for developing the domain knowledge for requirements elicitation [3], [6]. Using existing structured knowledge is another solution to solve this problem. Several methods [8], [2] use Common Criteria (CC) as such knowledge.

Even when we can obtain domain knowledge, the knowledge is not always machine-readable. For example, CC documents are normally written in PDF format and we cannot sometimes access the texts in the documents due to the security settings in PDF.

We can find some public and machine-readable knowledge base such as CAPEC, CVE or CWE. Common Vulnerabilities and Exposures (CVE) is a dictionary of publicly known information security vulnerabilities and exposures. Common Weakness Enumeration (CWE) is a set of software weakness. Although all of them seem to be useful for eliciting security requirements, we first use CAPEC in our research.

3 Method for Eliciting Security Requirements

The goal of this method is to support security requirements elicitation using CAPEC, which is structured and machine-readable knowledge resources. Although security experts should examine elicited security requirements after applying this method, this method enables them to save effort. We mainly explain our method in this section. In addition to CAPEC knowledge, we have to prepare a mapping between noun phrases in requirements descriptions and technical terms used in CAPEC. We also explain how to prepare such a mapping.

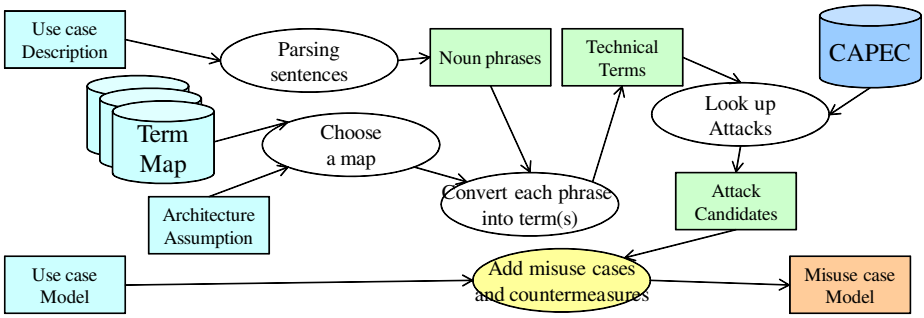

Fig. 1. Data flows among this elicitation method

3.1 Elicitation Method

Figure 1 shows the data flows among our requirements elicitation method. Boxes and cylinders correspond to data stores, ovals correspond to processes and arrows correspond to data flows in the figure. The inputs of the method are as follows.

- Use case description: In the method, we assume requirements are represented in a use case model and a use case description specifies each use case in natural language sentences. Security requirements elicitation is then performed for each use case. In this sense, a use case description is the main input of the method.
- Term maps: As mentioned at the beginning of this section, we have to prepare a mapping between noun phrases in a use case description and technical terms used in CAPEC because technical terms such as "SQL" or "buffer" are not directly used in use case descriptions. We thus have to prepare the mapping to bridge the gap between the noun phrases and technical terms. We call such a mapping "term map" in this paper. Because technical terms usually depend on a specific architecture or implementation, we prepare the mapping for each architecture assumption.
- Architecture assumption: Even at the requirements elicitation stage, some architecture assumptions such as client-server or cloud platform are considered. Such architecture assumptions give large effects on security issues because actual attacks and their protections are usually achieved on the basis of such assumptions. We assume such architecture assumptions are provided before requirements elicitation. Such assumptions correspond to an element "Technical_Context" in an attack pattern in CAPEC.
- Use case model: Because the output of the method is misuse case model [9] or MASG [7], we prepare a use case model as one of inputs.

The output is a misuse case model as mentioned above. The following elements in the model are elicited on the basis of an attack pattern.

- Misuse cases: An attack pattern itself directly specifies a misuse case.
- Countermeasure(s): In a misuse case model, a countermeasure is represented as a use case, and there exists a mitigate- or avoid-relationship between the use case and a misuse case. Countermeasures can be found in an element "Solutions_and_Mitigations" in CAPEC an attack pattern.

In the case of MASG [7], we explicitly specify assets (data) in use case model. CAPEC helps us to identify such assets.

The method in Figure 1 is repeatedly applied to each use case in a use case model. For each iteration, a different the use case description is used according to a use case, and the use case model is updated because misuse cases and countermeasures have been added during the former iterations. The processes in the figure are as follows.

- Parsing sentences: According to an article [1], noun phrases in a document play an important role for characterizing the domain of the document. We thus parse the sentences in a use case description, and pick noun phrases up as shown in Figure 1.
- Choose a map: We assume we have already had libraries of term maps for several architecture assumptions. We simply choose one of them on the basis of the description of the architecture assumption. We expect such a description is provided in the same form of "Technical_Context" element in CAPEC.
- Convert each phrase into term(s): As mentioned above, technical terms used in CAPEC are not usually used in use case descriptions. We thus convert each noun phrase into technical terms on the basis of a term map.
- Look up attacks: We look up attack patterns on the basis of technical terms. We assume attacks in the patterns can be applied to a use case specified in the use case description. We thus call the looked up attack patterns "attack candidates".
- Add misuse cases and countermeasures: For each attack candidate, we have to manually examine whether it can be really applied to the use case description. If it can be applied, we add misuse cases and their countermeasures to the use case model on the basis of the candidate.

Except "Add misuse cases and countermeasures", processes above can be performed automatically.

3.2 Preparing Term Maps

A term map is used for bridging the gap between phrases in requirements descriptions and technical terms in CAPEC. As we stated above, technical terms used in CAPEC are not usually used in use case descriptions. We thus have to bridge the gap between phrases in use case descriptions and technical terms by using the term map. Figure 2 shows an example of a term map for web applications and its usage. In this example, we have an architecture assumption that a system will be implemented as web application. We thus use this term map. At the left top of this figure, a part of use case description is shown, and two phrases "data" and "display" are focused. These phrases are converted into four technical terms "SQL", "schema", "database", "web page". These terms are used for finding attack patterns in CAPEC.

Although term maps should be manually gathered, they will be reused and improved step by step during their usages. We had experiences to gather this kind of domain knowledge [5], and we have empirically validated that we could improve the quality of such knowledge [4]. We thus assume we can also gather and improve the term maps in the same way.

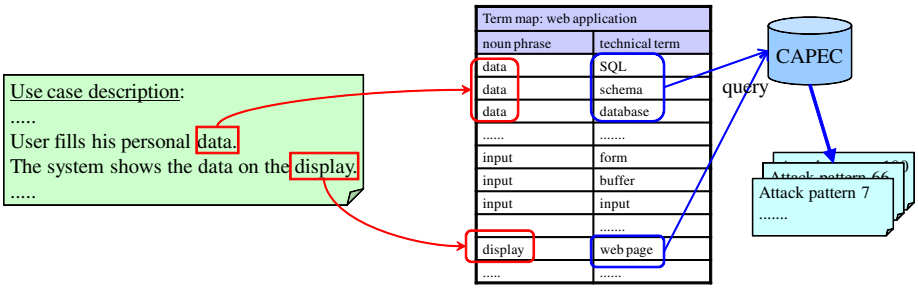

Fig. 2. An example of a term map for Web applications and its usage

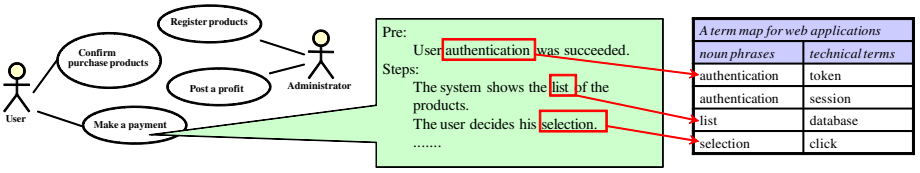

Fig. 3. A use case model, a use case description and a term map for web applications

3.3 Example of Applying the Method

We show a small example of applying the method. The example is about a web application for shopping. In figure 3, we show the use case model, a use case description and a term map used in this example. Because the system is implemented as a web application, a term map for web applications is used. In a use case description of a use case "Make a payment", three noun phrases "authentication", "list" and "selection" are identified. On the basis of the term map in Figure 3, these noun phrases are converted into four technical terms "token", "session", "database" and "click". Attack_Prerequisites of the following attack patterns contain each technical term as follows.

- token 4 patterns:
  - 102 Session Sidejacking
  - 36 Using Unpublished Web Service APIs
  - 39 Manipulating Opaque Client-based Data Tokens
  - 461 Web Services API Signature Forgery Leveraging Hash Function Extension Weakness
- session 11 patterns: 102, 103, 196, 21, 222, 226, 462, 467, 59, 60, 61
- database 3 patterns: 108, 109, 470
- click 2 patterns: 103, 222

Basically, all the patterns above become candidates of attacks. When we have to prioritize the attacks, we can refer to the values of other elements such as "Pattern_Completeness", "CIA_Impact" and "Typical_Severity".

4 Conclusion

In this paper, we proposed a method for eliciting security requirements. Because machine-readable and up-to-date security knowledge called CAPEC is used in the method, this method can reduce the effort of its user (normally requirements analysts).

Currently, we only use knowledge in CAPEC, but related machine-readable knowledge related to CAPEC such as CVE and CWE is also available. We will use such knowledge sources in addition to CAPEC. Although term maps seem to be useful for bridging the gaps between requirements descriptions and technical terms, we do not validate we can gather and improve the term maps step by step. We will continue to apply the method to various kinds of requirements descriptions to validate this point.

References

1. Capobianco, G., Lucia, A.D., Oliveto, R., Panichella, A., Panichella, S.: On the role of the nouns in ir-based traceability recovery. In: ICPC, pp. 148–157 (2009)
2. Houmb, S.H., Islam, S., Knauss, E., Jürjens, J., Schneider, K.: Eliciting security requirements and tracing them to design: An integration of common criteria, heuristics, and UMLsec. Requirements Engineering 15(1), 63–93 (2010)
3. Kaiya, H., Shimizu, Y., Yasui, H., Kaijiri, K., Saeki, M.: Enhancing domain knowledge for requirements elicitation with web mining. In: APSEC, pp. 3–12 (2010)
4. Kaiya, H., Suzuki, S., Ogawa, T., Tanigawa, M., Umemura, M., Kaijiri, K.: Spectrum analysis for software quality requirements using analyses records. In: COMPSAC Workshops, pp. 500–503 (2011)
5. Kaiya, H., Tanigawa, M., Suzuki, S., Sato, T., Kaijiri, K.: Spectrum analysis for quality requirements by using a term-characteristics map. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 546–560. Springer, Heidelberg (2009)
6. Kitamura, M., Hasegawa, R., Kaiya, H., Saeki, M.: A Supporting Tool for Requirements Elicitation Using a Domain Ontology. In: Filipe, J., Shishkov, B., Helfert, M., Maciaszek, L.A. (eds.) ICSOFT/ENASE 2007. CCIS, vol. 22, pp. 128–140. Springer, Heidelberg (2008)
7. Okubo, T., Taguchi, K., Yoshioka, N.: Misuse cases + assets + security goals. In: CSE, vol. (3), pp. 424–429 (2009)
8. Saeki, M., Hayashi, S., Kaiya, H.: Enhancing goal-oriented security requirements analysis using common criteria-based knowledge. International Journal of Software Engineering and Knowledge Engineering 23(5), 695–720 (2013)
9. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requir. Eng. 10(1), 34–44 (2005)
10. Zhao, Y., Dong, J., Peng, T.: Ontology classification for semantic-web-based software engineering. IEEE Transactions on Services Computing 2, 303–317 (2009)