Automated Software Security Requirements Recommendation Based on FT-SR Model

Jiangjuan Wang[1,3], Xiaohong Li[1,3], Zhiyong Feng[2,3], Jianye Hao[2,3], Guangquan Xu[1,3], Zhuobing Han[1,3]
[1]School of Computer Science and Technology, Tianjin University,
[2]School of Computer Software, Tianjin University,
[3]Tianjin Key Laboratory of Advanced Networking, Tianjin, China
Email:{jiangjuanwang, xiaohongli, zyfeng, jianye.hao, losin, zhuobinghan}@tju.edu.cn

***Abstract*—Since security is recommended to be evaluated at the beginning of the software development process, specifying software security requirements is inevitable in developing Critical Information Security Systems. However, according to the ISO/IEC 15408 (known as Common Criteria), determining detailed software security requirements (SRs) is quite challenging, complex which needs lots of expert knowledge of security. In this paper, a data-driven Functionality Topic-Security Requirement (FT-SR) model is proposed to recommend software SRs based on the relationship between software functionality specification and SRs from software Security Target (ST) which have been written in accordance with ISO/IEC 15408. First, we extract descriptions of functionality and tag SRs all from software STs. Second, Latent Dirichlet Allocation (LDA) is adopted to build functionality topics for product functionality description. Third, a FT-SR model is developed based on the mapping between product functionality topics and SRs which have been tagged in ST documents. Finally, a recommendation strategy is proposed to recommend SRs based on the FT-SR model for software products. Our experiments are performed on ST documents of over 600 software products provided by Common Criteria. Experimental results show that the proposed approach can generate a set of recommended SRs reducing the difficulty of SRs recommending even for people lack knowledge of security.**

Keywords-Security Requirements; the Common Criteria; Software Functional Description; FT-SR Model

I. Introduction

Many security issues occur in software as a result of errors and misspecifications in analysis, design and implementation [1]. Nowadays developers are more concerned about software security in the software development process. In order to develop security software products, specifying SRs is vitally important [2]. Security requirements of software systems can be challenging to identify during the requirements engineering process [3]. So far many methods have been proposed for SRs elicitation reference. Early work suggests that use case has been widely used during requirements for eliciting security threats and requirements [4]. As a result, use cases [4], misuse cases [5] and abuse cases [6] have all been proposed for specifying SRs, respectively. Although these methods are useful, they are inefficient and time-consuming because of without the guidance. Maria Riaz, et al. [7] developed a tool-assisted process presenting a list of applicable SRs templates for identifying SRs. And, they prove that the automatically-suggested templates usage increases the efficiency of identifying security requirements by comparing to a manual approach without the guidance of templates [3]. These researches all need using standards such as Common Criteria (referred to as "CC" in this paper), known as ISO/IEC 15408 [8]–[10]. The standard allows to select suitable SRs for a specific product or a group of products. However, since CC standards are too confusing to understand, these researches about determining softwares SRs processes are difficult and complex.

In this paper, we propose a new research direction of identifying and ensuring SRs for software products which only have functional descriptions without SRs. We firstly extract text description related to software functionality and tag related SRs all from software ST documents. ST documents are written in accordance with CC and certified by Competent and Independent Licensed Laboratories. Second, Latent Dirichlet Allocation (LDA) is used to build functionality topic feature for functional description of the evaluated software products. Meanwhile, according to the relationship between software functionality and SRs showing in STs, a FT-SR model is generated for software products. Finally, for given software products which only have functional description, SRs are recommended on the basis of the FT-SR model. A recommended strategy combining with a filtering framework [11] is developed to recommend SRs.

Therefore, the main contribution of this paper is as follows:

- A novel method is proposed to obtain softwares SRs even if person has little knowledge of security.
- The Functionality Topic-Security Requirement (FT-SR) model is built based on the mapping between functional description of software products and software products security requirements.
- A data-driven measure based on text description is considered to solve software security problems.

The remainder of the paper is organized as follows. Section 2 describes the methodology of overview. Section 3 analyzes the proposed method conducted in detail. Section 4 presents the experimental results and discussion. Section 5 concludes the paper and points out future work.

II. Method Overview

A. Problem Definition

As paying attention to software SRs and the relationship between functional description and SRs, we can formally

DOI reference number: 10.18293/SEKE2017-040

define SRs and the relationship respectively in order like:

$$SR_i = \{SFR_i, SAR_i\} = \{< sfc_{i,1}, \ldots, sfc_{i,K} >, < sac_{i,1}, \ldots, sac_{i,P} >\} \quad (1)$$

Note that, CC defines SRs as two aspects: security functional requirements (SFR) [9] and security assurance requirements (SAR) [10]. SFR_i and SAR_i respectively refer to security functional requirements and security assurance requirements which are SRs SR_i of software product S_i needed. $sfc_{i,j}$ represents security functional component. $sac_{i,j}$ represents security assurance component. K is the number of security functional components and P is the number of security assurance components in Common Criteria.

$$s_i = \{F_i, SR_i\} = \{< tp_{i,1}, \ldots, tp_{i,N} >, < sfc_{i,1}, \ldots, sfc_{i,K} >, < sac_{i,1}, \ldots, sac_{i,P} >\} \quad (2)$$

Note that: $F_i =< tp_{i,1}, \ldots, tp_{i,N} >$ refers to functionality feature of software product s_i , N is the total number of functionality topic for all software products, $0 \leq tp_{i,j} \leq 1$ represents the probability that software functionality is relevant to a certain functionality topic.

Accordingly, the problem of security requirements based on software functionality topic are defined as:

Q1: given the software product s_i and functional description from STs, how to generate the functionality topic feature and recommend its security requirements?

Q2: given the software product s_i, functional description and the recommendation security requirements, how to evaluate the recommendation results in our experiment?

B. Framework Overview

In order to solve those questions mentioned above, we introduce the architecture of our method which combines the FT-SR model and a recommendation strategy to recommend SRs for software products only having functionality requirements without SRs. It is shown in Fig.1.

Software Functional Description Gathering and Security Requirements Tagging: The goal of this process is to gather functional description and tag SRs from ST documents. Detail will be shown in Section 3.1.

Software Functionality Topic Clustering: The goal for this process is to obtain software functionality topic feature for all software products based on their functional descriptions. Here, we use LDA to cluster functional description being gathered and generate functionality topic features. Detail will be shown in Section 3.2.

Software Functionality Topics and Security Requirements (FT-SR) Model Building: For each software product, we have tagged its security requirements in accordance with its ST. We build functional topic-security requirements (FT-SR) model based on the mapping between its functional topics and its SRs. Detail will be shown in Section 3.3.

Security Requirements Recommending: For software product s_i which only have functionality requirements, we can get its SRs based on the functionality topics feature and FT-SR model combining a recommendation strategy. Detail will be shown in Section 3.4.

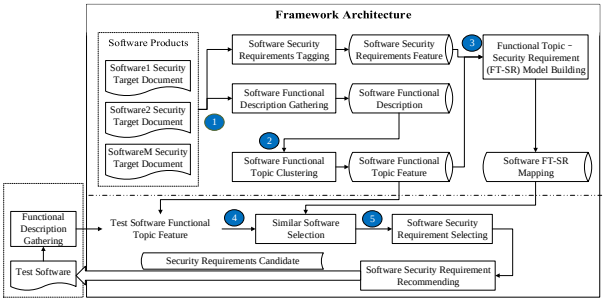

Fig. 1: Architecture of FTM-based Security Requirements Recommendation

III. The Proposed Method

This section discusses the methodology that has been used in the work.

A. Software Functional Description Gathering and Security Requirements Tagging

In this section, the descriptions of software functionality are extracted and security requirements are tagged all from Security Target for software products. Those software products are publicly available on the Common Criteria (http://www.commoncriteriaportal.org/products/). The ST documents are involved in more than 1000 products which have been certified by Competent and Independent Licensed Laboratories. ST documents have been downloaded and collected in this paper. Our task is to carefully read software product's ST documents to extract the description of functionality and tag SRs. For each software product, we extract its functional descriptions from the Chapter 1 of ST documents, meanwhile, we tag its security requirements in accordance with the Chapter 6 of ST.

B. Software Functional Topic Clustering

The goal for this process is to generate functionality topics for all the software products based on their functional descriptions. We follow the methodology using the Latent Dirichlet Allocation (LDA) to clustering all software's functional descriptions into different functionality topics, so that we can get the probability that each software product is relevant to certain functionality topics. Note that the summation of all the elements for each topic feature vector equals to 1. The results of LDA are N functional topics and the probability that each software product is relevant to the N functionality topics. Thus, functionality of all software products is represented as a topic feature vector, as follows:

$$F = \begin{bmatrix} tp_{1,1} & tp_{1,2} & \ldots & tp_{1,N} \\ tp_{2,1} & tp_{2,1} & \ldots & tp_{2,N} \\ \ldots & \ldots & \ldots & \ldots \\ tp_{M,1} & tp_{M,1} & \ldots & tp_{M,N} \end{bmatrix} \quad (3)$$

Where: $tp_{i,j}$ refers to the probability that one software product is relevant to one functional topic, $0 \leq tp_{i,j} \leq 1$. $F_i =< tp_{i,1}, \ldots, tp_{i,N} >$ refers to the set of probability that software product s_i is relevant to each functional topics. N is the number of all functionality topics that all software products were clustered into different functionality topics; M is the number of all software products.

In this section, the correlation which also is functionality topic feature between functional description and each functionality topic can be obtained for each software product.

C. Functional Topic-Security Requirements (FT-SR) Model

For each software product, SRs are tagged in section 3.1. And, the FT-SR model is built between the relationship between functionality topics feature, software product and SRs for software products. First, the tagged SRs feature is presented a feature vector, as follow:

$$SR = \begin{bmatrix} sfc_{1,1} & \ldots & sfc_{1,K}, & sac_{1,1} & \ldots & sac_{1,P} \\ sfc_{2,1} & \ldots & sfc_{2,K}, & sac_{2,1} & \ldots & sac_{2,P} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ sfc_{M,1} & \ldots & sfc_{M,K}, & sac_{M,1} & \ldots & sac_{M,P} \end{bmatrix} \tag{4}$$

Where $sfc_{i,j}$ represents specific security functional requirements sfc_j what software product s_i needed. K is the total security functional components what CC have defined. Note that, if $sfc_{i,j} = 0$, this means the software product s_i doesn't use security functional component sfc_j.

Where $sac_{i,j}$ represents the specific security assurance requirements sac_j what software product s_i needed. P is the total security assurance components what CC have defined. Note that, if $sac_{i,j} = 0$, this means the software product s_i doesn't use security assurance component sac_j.

According to above steps, we build the FT-SR model which is the relationship between functionality topic and SRs for each software product. The relationship is expressed as same as (2).

D. Security Requirements Recommending

Software Products of Similar Functionalities Selecting.

For a given software product s_i which have only functional descriptions, as LDA is used to mapping its functional description into generate probability of each functionality topics, software product s_i is represented as a functionality topic feature vector. Therefore, this paper calculate the similarity between two software products s_i, s_j as follow:

$$Sim(s_i, s_j) = \frac{\sum_{n=1}^{N} tp_{i,n} \times tp_{j,n}}{\sqrt{(\sum_{n=1}^{N} tp_{i,n}^2)(\sum_{n=1}^{N} tp_{j,n}^2)}} \tag{5}$$

Thus, we can get a list which related software products with similarity functionality to the given software product s_i .Using a subset of software products for recommendation can gain a better performance because of the noise in the dataset. Therefore, we use the threshold method to obtain a subset of appropriate software products with similarity larger than the given threshold θ_{sim}. Given the selected software products $P^*(s_i)$ with similarity larger than the given threshold θ_{sim}. The threshold is changing and the better threshold has been found by comparing the assurance of recommendation result in our experiment.

Security Requirements Selecting and Recommending.

According to the selected software products $P^*(s_i)$, we can calculate the likelihood that software product s_i need security requirements SR_j is calculated as follow:

$$l(s_i, SR_{i,j}) = \frac{\sum_{s_k \in P^*(s_i)} Sim(s_i, s_k) l(s_k, SR_{k,j})}{\sum_{s_k \in P^*(s_i)} Sim(s_i, s_k)} \tag{6}$$

Note that, if $l(s_k, SR_{k,j})$=1 , software product s_k needs the security requirement SR_j, otherwise $l(s_k, SR_{k,j})$=0. Thus, we select the likelihood vector for the given software products $L(s_i) =< l(s_i, sfc_{i,1}), \ldots, l(s_i, sfc_{i,Q}), l(s_i, sac_{i,1}), \ldots, l(s_i, sac_{i,R}) >$, where $l(s_i, sfc_{i,j}) \geq l(s_i, sfc_{i,k}) \geq \theta_{NP}$,$1 \leq j \leq k \leq Q$, $l(s_i, sac_{i,j}) \geq l(s_i, sac_{i,k}) \geq \theta_{NP}$, $1 \leq j \leq k \leq R$, θ_{NP} is the likelihood threshold.

Therefore, the set of $SR_i =< sfc_{i,1}, \ldots, sfc_{i,Q}, sac_{i,1}, \ldots, sac_{i,R} >$ is the recommendation of security requirements candidate for software product s_i.

IV. Experimental Results and Evaluation

A. Data Set

Since Common Criteria is well-known for Information Technology Security Evaluation and most software products are evaluated by Common Criteria, we use the "Certified Products" as dataset from http://www.commoncriteriaportal.org/products/. The software products have been classified into 14 categories by Competent and Independent Licensed Laboratories. Our method applies to the 14 categories of software products which behave in the exact similar way. Considering the number of software products each category, we have mainly gathered three categories of products which are ICs, Smart Cards and Smart Card-Related Devices and Systems, Multi-Function Devices and Network and Network-Related Devices and Systems because of their large quantity. Then as software products have grown into different versions, we choose the latest version of every product and download its Security Target separately in accordance with categories. Overview of datasets is shown in Table 1.

TABLE I: Overview of Datasets

Dataset	Number
ICs, Smart Cards and Smart Card-Related Devices and Systems	403
Multi-Function Devices	115
Network and Network-Related Devices and Systems	138

B. Result and Evaluation

In our method, we use the similarity threshold θ_{sim} to select the software products which have similar functionalities. Therefore, in order to evaluate its assurance, we randomly select 25% software products from each category products as the testing dataset respectively and the three experiments

reported in Figure 2. We vary threshold θ_{sim} from 0 to 1 to generate different recommendations. Mean Average Precision (MAP) are used to evaluate the performance of accuracy for results of recommendation and answer the question 2 mentioned above. It can be formally defined as follow:

$$MAP = \frac{1}{R}\sum_{r=1}^{R}\frac{I_r}{N_r} \tag{7}$$

Where R refers to the number of software products in testing dataset, N_r refers to the number actually used security requirements for rth product s_r. I_r refers to the number of the recommended security requirements which belong to the actually used security requirements N_r. And, the MAP of the recommendation is reported in Fig 2. As shown in Fig 2, it can be seen that for the dataset, if $\theta_{sim} > 0.5$, the MAP is decreasing; This is because that with a too larger similarity threshold, most of software products can not have enough similar products to generate a valid recommendation. Therefore, in the rest of this paper, we set $\theta_{sim} = 0.5$ for the selection. For the dataset, those software products are classified into 14 categories. Thus, for the given software products, its type should also belong to the 14 categories for recommendations.

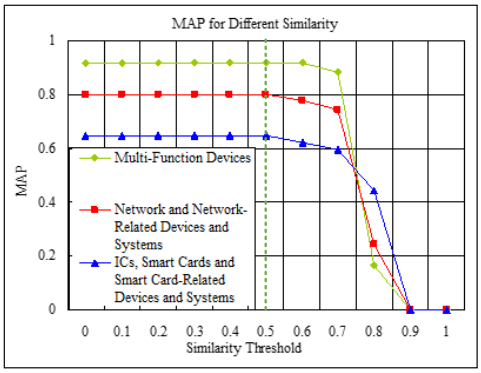

Fig. 2: Overview of Datasets

C. Comparison

Now, Common Criteria is used to analyze security requirements internationally. This process specifies [12], among other things, (a) which are the assets to be protected (b) which are the threats to the declared assets and (d) which are the countermeasures. The CC then describes the countermeasures and demonstrates that they are sufficient to counter the declared threats. Therefore, if developers want to get SRs for software products, they have to acquire the knowledge of CC and find out what kind of information they needed in the procedures. So according to Common Criteria security standard, obtaining softwares SRs process is an extremely difficult and complex task. In addition, its results are largely influenced by subjective knowledge of analyst, so accuracy of security requirements can't be guaranteed.

However, our method solves those disadvantages showing above. For given software products which only have functional requirements without SRs, we could recommend its SRs applying on their functional descriptions even for people with little knowledge of security engineering and Common Criteria.

V. Conclusion

Identifying SRs is an important part during the software development. Most of existing work is related to Common Criteria. Since Common Criteria standards are often too confusing to understand, these researches are difficult and complex. In this paper, functionality of software products are considered to recommend SRs. Functional descriptions are extracted manually and clustered into functionality topic. Basing on the mapping between SRs and functionality topics, the FT-SR model is built. Finally, for given software products which only have functional requirements without SRs, their SRs are recommended based on FT-SR model. The advantage of our method is acquiring SRs even for people lack knowledge about security.

In the future, we will further extend our method to more precisely recommend SRs of software products to help SRs identification. We can consider more software information to recommend more appropriate SRs.

VI. Acknowledgement

This work has partially been sponsored by the National Science Foundation of China (No. 61572349, 61272106).

References

[1] U. Erlingsson, "Data-driven software security: Models and methods," in *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*. IEEE, 2016, pp. 9–15.

[2] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer standards & interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[3] M. Riaz, J. Slankas, J. King, and L. Williams, "Using templates to elicit implied security requirements from functional requirements-a controlled experiment," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2014, p. 22.

[4] M. S. Ware, J. B. Bowles, and C. M. Eastman, "Using the common criteria to elicit security requirements with use cases," in *SoutheastCon, 2006. Proceedings of the IEEE*. IEEE, 2005, pp. 273–278.

[5] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements engineering*, vol. 10, no. 1, pp. 34–44, 2005.

[6] J. Mcdermott, "Abuse-case-based assurance arguments," in *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings*, 2001, pp. 366–374.

[7] M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*. IEEE, 2014, pp. 183–192.

[8] "Common criteria for information technology security evaluation (version 3.1, revision 4) part 1: Introduction and general model," *ISO/IEC 15408*, 2012.

[9] "Common criteria for information technology security evaluation (version 3.1, revision 4) part 2: Security functional components," *ISO/IEC 15408*, 2012.

[10] "Common criteria for information technology security evaluation (version 3.1, revision 4) part 3: Security assurance components," *ISO/IEC 15408*, 2012.

[11] K. Huang, J. Han, S. Chen, and Z. Feng, *A Skewness-Based Framework for Mobile App Permission Recommendation and Risk Evaluation*. Springer International Publishing, 2016.

[12] M. Saeki and H. Kaiya, *Security Requirements Elicitation Using Method Weaving and Common Criteria*. Springer Berlin Heidelberg, 2009.