Finding Potential Threats in Several Security Targets for Eliciting Security Requirements

Haruhiko Kaiya
Kanagawa University
Hiratsuka, Japan
Email: kaiya@kanagawa-u.ac.jp

Shinpei Ogata
Shinshu University
Nagano, Japan
Email: ogata@cs.shinshu-u.ac.jp

Shinpei Hayashi
and Motoshi Saeki
Tokyo Institute of Technology
Tokyo, Japan
Email: {hayashi,saeki}@se.cs.titech.ac.jp

Takao Okubo
Institute of Information Securiry
(IISEC) Yokohama, Japan
okubo@iisec.ac.jp

Nobukazu Yoshioka
National Institue of Informatics
(NII) Tokyo, Japan
nobukazu@nii.ac.jp

Hironori Washizaki
Waseda University
Tokyo, Japan
washizaki@waseda.jp

Atsuo Hazeyama
Tokyo Gakugei University
Tokyo, Japan
hazeyama@u-gakugei.ac.jp

***Abstract*—Threats to existing systems help requirements analysts to elicit security requirements for a new system similar to such systems because security requirements specify how to protect the system against threats and similar systems require similar means for protection. We propose a method of finding potential threats that can be used for eliciting security requirements for such a system. The method enables analysts to find additional security requirements when they have already elicited one or a few threats. The potential threats are derived from several security targets (STs) in the Common Criteria. An ST contains knowledge related to security requirements such as threats and objectives. It also contains their explicit relationships. In addition, individual objectives are explicitly related to the set of means for protection, which are commonly used in any STs. Because we focus on such means to find potential threats, our method can be applied to STs written in any languages, such as English or French. We applied and evaluated our method to three different domains. In our evaluation, we enumerated all threat pairs in each domain. We then predicted whether a threat and another in each pair respectively threaten the same requirement according to the method. The recall of the prediction was more than 70% and the precision was 20 to 40% in three domains.**

Keywords–Security Requirements Analysis; Requirements Elicitation; Common Criteria; Security Target; Domain Knowledge.

I. INTRODUCTION

Knowledge about computer security is important for requirements elicitation because security has effects on development costs and efforts. Although we expect that security experts will provide such knowledge, they cannot always do so. Researchers thus developed methods of eliciting requirements using documented knowledge [1] [2] [3] [4]. In such methods, the method helps a requirements analyst to find new security requirements on the basis of requirements already elicited. However, there are a few methods of developing or acquiring such knowledge [5] [6].

It was not easy to acquire such security knowledge that are high quality because security experts tacitly held the knowledge and they rarely document it. There have recently been several structured documents that have been of high quality where such knowledge has explicitly been represented. Examples are the Security Target (ST) in Common Criteria (CC) [7] and Common Attack Patterns Enumeration and Classification (CAPEC) [8]. Each element in a document is uniquely identified in such documents, and the relationships between the elements are formally specified. Existing methods of developing documented knowledge did not fully utilise the explicit structure of knowledge sources, but they simply used linguistic characteristics in such documents with the help of lightweight natural language processing (NLP).

Saeki et al. [9] reported that using more than two knowledge sources contributed to comprehensively eliciting security requirements. Especially, when a threat in a ST and similar threat in another ST were together examined in our current requirements analyses, security requirements could be elicited more comprehensively than ever. However, it took a huge amount of effort because useful knowledge was scattered over several different sources, and requirements analysts had to manually find them step by step. Therefore, such sources have to be integrated so that the analysts can efficiently and comprehensively find potential threats protected by security requirements. However, no one cannot know a threat and another will threaten the same requirement without examining their contents.

We then set up three research questions.

- RQ1: How to integrate several structured security documents systematically so that requirements analysts can elicit security requirements comprehensively and efficiently?
- RQ2: How to use security documents written in different languages, such as English and French?
- RQ3: How to perform such integration without the knowledge whether one threat and another will respectively threaten the same requirement?

The contribution of this paper is to provide answers to research questions above.

We propose a method of integrating several STs in this paper for three main reasons. First, ST provides highly structured documents that are useful for semantically integrating them. Second, the documents are provided in machine-readable

 ISBN: 978-1-61208-432-9

format. Third, STs refer to issues related to requirements although most security related documents refer to design and/or implementation issues.

We assume that analysts first find partial threats to a system by themselves, and the integrated STs contain them. Countermeasures to the threats are candidates for the security requirements in the system in our method. Our method then recommends the other threats to be examined for eliciting additional security requirements. It is very important to find potential candidates for threats when security requirements are being elicited because threats that are not taken into account make the system vulnerable. The integrated STs thus contribute to improving the quality of eliciting security requirements because they provide as many candidates as possible.

The relationships between a threat and other threats should be systematically identified to find such additional threats. Various kinds of semantic relationships such as similarities and dependencies between threats are useful. We focus on the means for protection against each threat to identify these relationships. Such means are called security functional requirements (SFRs) in CC. We assume several SFRs are commonly used if such semantic relationships between a threat and another are established. Our method is based on this assumption. The assumption is confirmed and explained in Section V.

The rest of this paper is organised as follows. The next section reviews related work on methods of eliciting requirements using reusable knowledge, and methods of developing such knowledge. Section III introduces CC, which provide structured knowledge on security, and methods using that knowledge. Section IV presents a method of integrating several STs. We also present its background, requirements and a discussion. We next explain our evaluation on whether our method worked well in Section V. We address the three research questions in summary and pose future issues.

II. Related Work

Structured knowledge, in general, such as ontology is widely used in the field of software engineering [10]. There are numerous methods of eliciting requirements using ontology in the field of requirements engineering [1] [2] [3] [4]. We expect knowledge such as ontology will contribute to the completeness and correctness of requirements, and such an expectation will only be satisfied when this knowledge is comprehensive enough. However, it is not easy to develop or acquire such knowledge in real situations. We thus have to investigate how to develop or acquire such knowledge.

Two types of different researches related to ontology exist. One is about designing the meta-model (syntax) of the ontology [11] [12]. Another is about developing the concrete instances of ontology. Our method in this paper belongs to the second type, and we simply use the structure of the ST as a meta-model of model instances.

Research on developing ontology in general already exists [13] [14] [15] [16], and most researchers have used NLP techniques. There are already a few knowledge integration methods and tools for requirements engineering [5] [6], and most of them also use NLP, and do not focus on highly structured documents. A method of developing security ontology was also proposed [17]. However, few automated or reused mechanisms were taken into account in the method.

Much structured knowledge is available in the field of security. However, most of it is not related to requirements but to design or implementation [18]. The ST in CC explained in detail below is one of a few exceptions because requirements related concepts such as threats and objectives are highlighted in it.

III. CC and Methods of Eliciting Requirements Using their Knowledge

CC represents an international standard that prescribes how to write documents to assess the security properties of information systems. CC consists of several structured documents and one of these is the ST. ST can be used to improve security requirements elicitation for the following reasons. First, we can find explicit relationships among threats to assets in a system, objectives to mitigate or avoid individual threats, and functionalities to implement the objectives, which are called SFRs. We can usually find the list of assets and implicit relationships between the assets and other elements above. Second, we can find already certified STs for individual IT products [7] and they are categorised on the basis of the types of the products.

Figure 1 outlines part of an ST for an Information Technology (IT) product for an Integrated Circuit (IC) card, where each threat and objective has its own unique name such as T.Skimming and O.Data_Conf, and each of them has its own explanation written in natural language sentences. The templates of SFRs are chosen from the catalogue provided by the CC framework (called CC Part 2) with several parameters, and each template is instantiated in each ST by assigning some values to the parameters. Each SFR has its own unique name such as FIA_UID or FCS_COP. The explicit relationships among threats, objectives and SFRs are provided in the ST so that each threat is countered by some concrete means, i.e., SFRs. The relationships between assets and the others are implicitly provided because no formal rules for naming assets and relating them to others are specified in CC. The reader of an ST should manually follow and understand issues related to assets.

Because the knowledge in STs is useful and it is easy for computers to operate them, as previously mentioned, there are already several methods of requirements elicitation [9] [19] [20] [21] using STs. In most of them, elements such as threats, objectives and SFRs are used for resources to elicit security requirements on the basis of existing functional requirements or goals.

We briefly introduce a method of eliciting security requirements using the knowledge in STs in a previous article [9]. Figure 2 outlines the flows of inputs and outputs with the method. The inputs are knowledge and functional requirements (FRs), and the outputs are security requirements (SRs). We can explain the steps in the method with this figure. We can also explain why integrating several STs is helpful in the method.

1) An analyst elicits or acquires functional requirements (FRs) in advance. There is “FR10” at the top right of Figure 2.
2) He/she has to explore assets in an FR or its threats by referring to descriptions of assets and threats in an ST. In this example, he focuses on the term “IC card” in the FR because an ST C229 contains an asset called an “IC chip”.

 ISBN: 978-1-61208-432-9

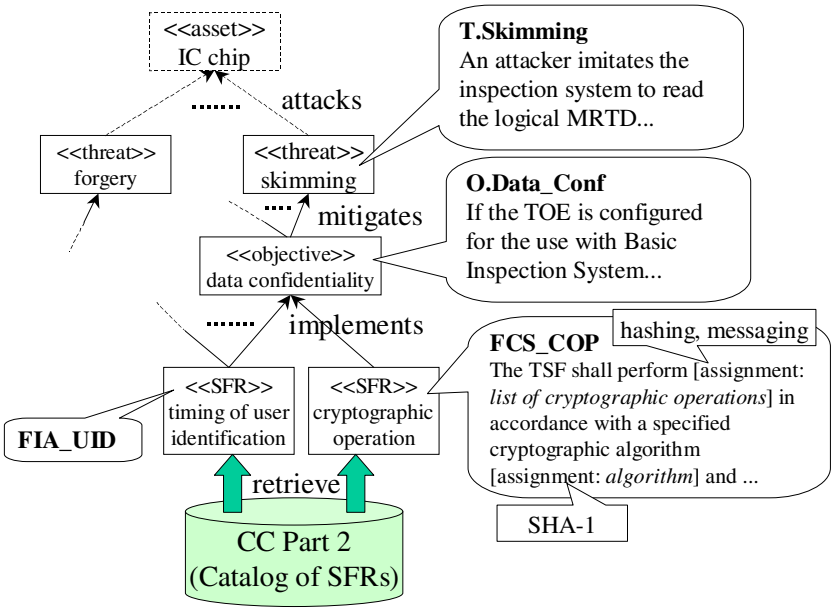


Figure 1. A part of ST for IC Card System

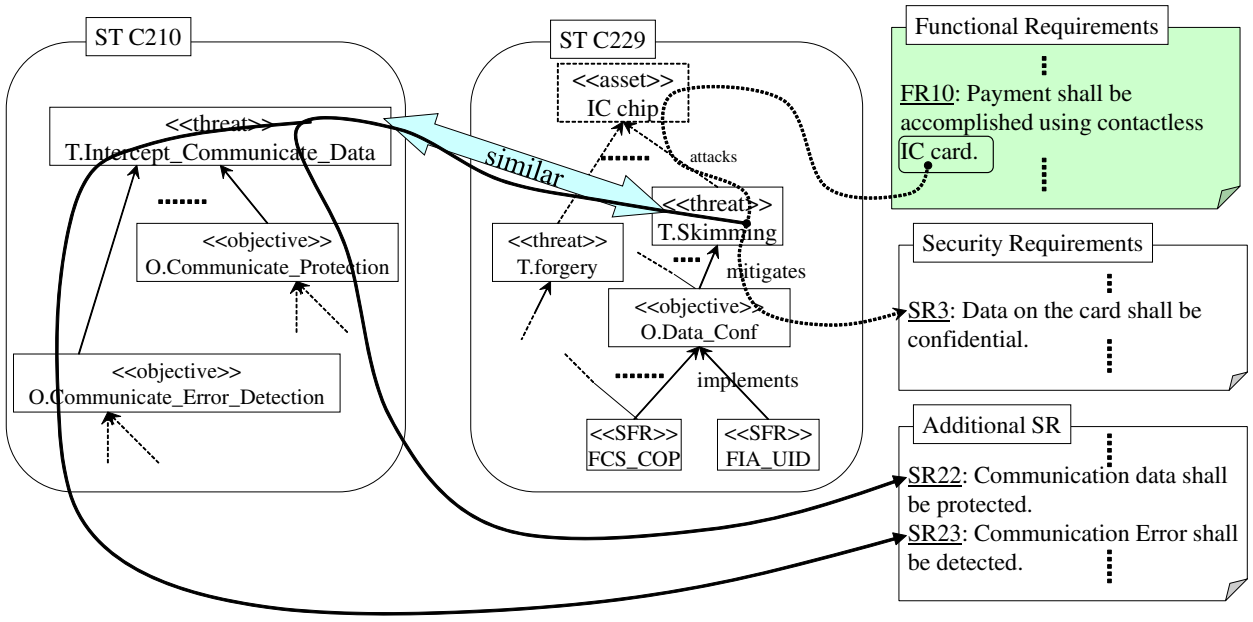


Figure 2. Example of security requirements elicitation using two STs

3) He/she then identifies threats to the asset, and regards objectives to the threats in the ST as candidates of security requirements (SRs).
In this example, an objective "O.Data_Conf" is added as "SR3" because the objective mitigates the threat "T.Skimming" and the threat threatens the asset "IC chip" in the ST. The dotted curved line in the figure indicates the trace in this step.
4) He/she finds threats in other STs that are similar to the threats that were originally identified. He/she then regards objectives to the threats in other STs as additional candidates of SRs.
He/she assumed "T.Skimming" in C229 in Figure 2 was similar to "T.Intercept_Communicatie_Data" in C210. He/she thus systematically finds SR22 and SR23 as seen in the figure. The curved lines in the figure indicate the traces in this step.
5) He/she repeats the steps above for each FR.

Step 4 plays a role in integrating several different STs, but how to integrate them is beyond the scope of Saeki et al. [9]. The main goal of the research discussed in this paper is to identify similarity among threats used in Step 4.

 ISBN: 978-1-61208-432-9

IV. SIMILAR THREATS DERIVED FROM SEVERAL STS

The goal of the method presented in this paper is to derive the pair of similar threats each of which threatens the same requirement from several STs. A pair of "T.Intercept_Communicate_Data" and "T.Skimming" in Figure 2 is an example of this pair. We predict such a pair on the basis of SFRs commonly mitigating or avoiding both threats. Even if several SFRs are common countermeasures to two threats, the threats are not always similar to each other. We thus need *a threshold* on such commonality so that we determine whether two threats with a certain commonality are similar or not. The threshold should be determined without the knowledge whether two threats are actually similar to each other. Metrics in Section IV-B are mainly used to explain how to determine this threshold, and how to derive pairs of threats is explained in Section IV-C.

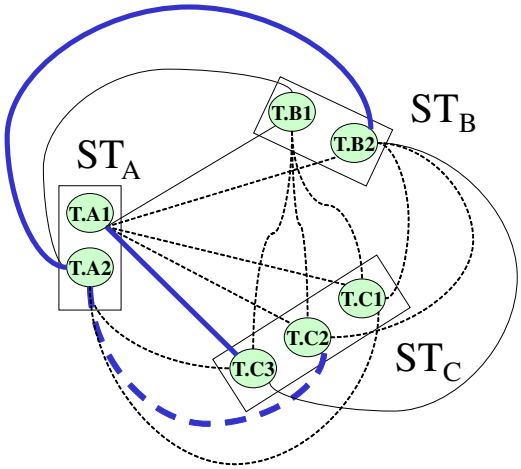
A. Background and Requirements for Deriving Threat Pairs

The method of eliciting security requirements in Section III uses relationships among existing FRs, attackers requirements (threats), countermeasures (objectives) and concrete means for implementing countermeasures (SFRs). Although relationships among threats, objectives and SFRs are explicitly represented in STs, requirements analysts should identify the relationships between FRs and others by themselves. Focusing on assets in individual FRs is one way to make such identifications, but we have to take into account synonyms are handled for the name of an asset. In addition, relationships between assets and threats should be manually investigated in an ST as was mentioned in Section III. Even though there are difficulties in identifying the relationships between FRs and others, an analyst has to first find one or a few threats, objectives or SFRs related to an FR.

Once one or a few of them can be found, the explicit structure in an ST like that in Figure 2 can be systematically utilized to expand the number of candidates of threats to each FR. Finding STs in the same application domain is not very difficult because they are categorised into domains on their web site [7]. Establishing a relationship between a threat in an ST and another threat in another ST is not very easy if we simply read their explanatory sentences. Although terms in an ST are normally used consistently within an ST, there is little consistency in terms between one ST and another, especially individual STs have been written by different technical people or different companies. In addition, STs can be written in different languages such as English, Japanese or French. In such cases, it is very difficult to integrate several different STs on the basis of terms in each ST.

B. Metrics related to Finding Threat Pairs

We define the following metrics for the method of deriving pairs of similar threats below. Let us use the example in Figure 3 to explain each metric. Each box in this figure corresponds to an ST, each circle corresponds to a threat, and each straight or dashed line specifies a potential pair explained below. The number of all potential pairs is 16. A legend for each kind of line can be found in the figure. In this example, threat pairs are derived from three STs. The STs are called $\mathrm{ST_A}, \mathrm{ST_B}, \mathrm{ST_C}$, where $\mathrm{ST_A}$ contains two threats, $\mathrm{ST_B}$ contains two threats, and $\mathrm{ST_C}$ contains three threats respectively.



Legend & summary

Correct \ Estimated	Yes	No
Yes	2 ———	1 - - -
No	3 ———	10 ----------

Figure 3. Example of three STs and threat pairs among them.

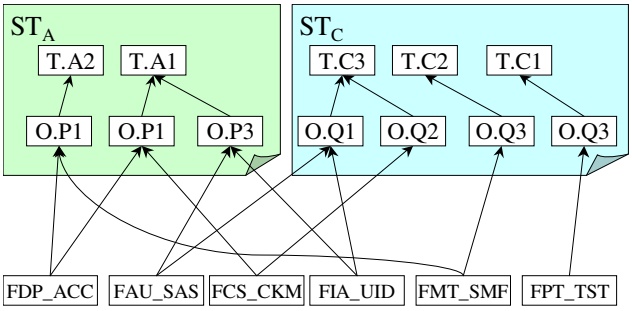


Figure 4. Example of two STs to explain metrics in Section IV-B

- *Potential pairs*:
  Let us focus on a pair of threats, each of which belongs to different STs. We call all such pairs *potential pairs*, and POT denotes the set of all potential pairs. There are 16 potential pairs, $|POT|$, in Figure 3. The number is derived on the basis of (1).

$$(\sum_{\mathrm{i=A}}^{\mathrm{C}}(\mathrm{nt}(\mathrm{ST_i}) \times ((\sum_{\mathrm{j=A}}^{\mathrm{C}}\mathrm{nt}(\mathrm{ST_j})) - \mathrm{nt}(\mathrm{ST_i}))))/2 \quad (1)$$

  In (1), $\mathrm{nt}(x)$ is a function to obtain the number of threats in x, which is the name of an ST. The equation (1) is instantiated in Figure 3 as follows because $\sum_{\mathrm{j=A}}^{\mathrm{C}}\mathrm{nt}(\mathrm{ST_j})$ is seven.

$$((2 \times (7-2)) + (2 \times (7-2)) + (3 \times (7-3)))/2$$

- *Correct pairs*:
  As was explained in the introduction, we expect threat pairs will help a requirements analyst to elicit additional security requirements when he/she finds one or a few threats threaten existing requirements. If both threats in a pair respectively threaten the same

requirement, we call the pair *a correct pair*, and the set of all correct pairs is represented as COR. The correct pairs are depicted by the thick line in Figure 3. Any correct pair is contained in the potential pairs, i.e., $COR \subseteq POT$. We cannot know COR in an actual situation because the number of POT is usually too huge to examine the meaning of each threat.

- *Estimated pairs*:
  Because no one knows the correct pairs, we have to predict whether a pair is a correct pair or not on the basis of computer-identifiable information. We used *SFR-commonality* in our method, which is explained below, to make this prediction. When SFR-comonality determines that a potential pair seems to be a correct pair, we call the potential pair *an estimated pair*, and the set of all estimated pairs is called EST. Any estimated pair is contained in potential pairs, i.e., $EST \subseteq POT$. Estimated pairs are represented by straight lines in Figure 3. The decision by SFR-commonality is not always correct. For example, there are five estimated pairs and only two out of them are correct in Figure 3. In total, 12 out of 16 pairs are correctly decided, but the others are not.
- *SFR-commonality*:
  This commonality is used to determine whether a threat in a ST is similar to another in another ST. It is a function with a threat pair as an argument. When the value is nearby one, two threats are similar to each other. When the value is nearby zero, they are not similar. We use the Jaccard index [22] to define SFR-commonality. The Jaccard index Jac returns the degree of commonality between two sets X and Y as shown in (2).

$$Jac(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (2)$$

  The returned degree of (2) takes a value from zero to one. To use the Jaccard index to construct SFR-commonality, we define the following function (3).

$$USFR : T \rightarrow 2^{ALLSFR} \quad (3)$$

  T in (3) is the set of threats and $ALLSFR$ is the set of all existing SFRs defined in CC Part II. Function $USFR$ returns the set of SFRs transitively used in a threat t. We will now explain how to calculate the commonality between two threats by using the example in Figure 4, in which two STs, threats, objectives and SFRs used in the STs are depicted. Intuitively, T.A1 and T.C3 are similar to each other because almost the same SFRs were used to avoid or mitigate threats. Commonality is defined here so that it represents such intuition.
  The results obtained by applying $USFR$ to the threats in Figure 4 are:

$USFR(\text{T.A2}) = \{\text{FDP\_ACC}, \text{FMT\_SMF}\}$
$USFR(\text{T.A1})$
$= \{\text{FDP\_ACC}, \text{FAU\_SAS}, \text{FCS\_CKM}, \text{FIA\_UID}\}$
$USFR(\text{T.C3})$
$= \{\text{FAU\_SAS}, \text{FCS\_CKM}, \text{FIA\_UID}\}$
$USFR(\text{T.C2}) = \{\text{FMT\_SMF}\}$
$USFR(\text{T.C1}) = \{\text{FPT\_TST}\}$

  Because each potential pair of threats p consists of two threats $t1$ and $t2$, SFR-commonality Com can be defined as shown in (4).

$$Com(p) = Jac(USFR(t1), USFR(t2)) \quad (4)$$

  In (4), p is $\{t1, t2\}$.
  Because whether a pair is an estimated pair depends on the threshold td mentioned above, the set of estimated pairs EST is parameterised by the threshold, i.e., $EST(td)$, and is defined as shown in (5).

$$EST(td) = \{x | x \in POT \wedge Com(x) \geq td\} \quad (5)$$

  In (5), POT is the set of all potential pairs.
  For example, we calculate SFR-commonality in Figure 4 as:

$Com(\{\text{T.A1}, \text{T.C1}\}) = 0$
$Com(\{\text{T.A1}, \text{T.C2}\}) = 0$
$Com(\{\text{T.A1}, \text{T.C3}\}) = 3/4 = 0.75$
$Com(\{\text{T.A2}, \text{T.C1}\}) = 0$
$Com(\{\text{T.A2}, \text{T.C2}\}) = 1/2 = 0.5$
$Com(\{\text{T.A2}, \text{T.C3}\}) = 0$

  We can then calculate $EST()$ as:

$EST(0.1)$ = { {T.A2, T.C2}, {T.A1, T.C3} }
$EST(0.5)$ = { {T.A2, T.C2}, {T.A1, T.C3} }
$EST(0.7)$ = { {T.A1, T.C3} }
$EST(0.9)$ = { }

- *Estimated gain*:
  We expect a threat pair will suggest an additional threat protected by additional security requirement(s) when a threat in the pair has already been identified. Therefore, the number of the threats used for finding security requirements has increased by the number of threat pairs. The estimated gain indicates such a degree, and is defined as in (6).

$$\frac{|EST| + |ALL|}{|ALL|} \quad (6)$$

  In (6), ALL is the set of all threats. In Figure 3, ALL is {T.A1, T.A2, T.B1, T.B2, T.C1, T.C2, T.C3}, and the estimated gain is about 1.7 (=((2+3)+(2+2+3))/(2+2+3)). We occasionally represent this rate as a percentage such as 170 %. When the estimated pairs are rigorously chosen, $|EST|$ becomes nearly zero. In that case, the estimated gain becomes nearly 100 %. We do not apply transitivity when we calculate gain. For example, the pair T.A1 and T.B2 in Figure 3 is a transitive estimated pair because the pair T.A1 and T.C3 is an estimated pair and the pair T.C3 and T.B2 is also. However, we do not apply transitivity because most threats can become pairs with such transitivity.
  Because we can calculate the estimated gain in actual cases, we expect that the estimated gain can be an indicator to define the SFR-commonality below. The reasons for the expectation are as follows. Because of $EST \subseteq POT$, the upper limit of the estimated gain can be known in advance, i.e., $\frac{|POT|+|ALL|}{|ALL|}$. Because $EST = POT$ is unrealistic, the estimated gain should not become this upper limit. When an ST contains a set of threats, another similar ST normally contains a similar set. The total number of threats, i.e., $|ALL|$,

thus influences the proper size of EST, i.e., the size of COR, because of the commonality of sets of threats between STs. We thus expect that the estimated gain will be an indicator to predict estimated pairs.

C. Method of deriving Threat Pairs

The goal of our method is to derive threat pairs from several STs so that requirements analysts can elicit security requirements against the potential threats. The main characteristics of this method are explained in what follows.

- The method enables us to use several STs written in different languages such as English and French together because STs use common SFRs.
- The method can be automatically executed except to determine the threshold. The threshold is used to determine whether one threat can be regarded to be similar to another.
- In the method, the change of estimated gain plays a role of a clue to determine the threshold. The change of estimated gain can be automatically calculated without knowledge as to whether two threats are actually similar to each other.

The six steps in the method are as follows. All steps except step 5 can be performed automatically.

1) We prepare several STs that belong to the same domain. Although only three STs were integrated in our evaluation, our method can accept any number of STs and there are a huge number of STs in the same domain. For example, we could find 653 STs in the IC card domain [7].
2) We enumerate POT in these STs.
3) We calculate the SFR-commonality for each potential pair. It takes a value from zero to one.
4) We have to define a threshold on SFR-commonality so that we determine EST. We call a candidate for such a threshold *the lower limit of commonality*. We present the changes in estimated gain along with progress in the lower limit of commonality.
5) We choose a lower limit as *the threshold*. The threshold should be subjectively determined, but the changes in the estimated gain along with the progress in the lower limit will help us to determine this.
   A typical way to determine the threshold is as follows. An analyst focuses on the estimated gain where the lower limit is zero. The estimated gain in this case has the largest value, i.e., $(|POT| + |ALL|)/|ALL|$. He/she then increases the lower limit step by step until the estimated gain becomes stable. The lower limit at the beginning of the stable range can be a threshold.
6) EST is determined by the threshold above. According to each threat pair in EST, a threat in the pair is recommended as resources to expand security requirements when a requirements analyst has already elicited one or a few security requirements protecting another threat in the pair.

D. Discussion

As was discussed in Section IV-A, we want to expand the candidates for threats to individual FRs. However, finding the first candidate is beyond the scope of this method.

We assume SFR-commonality is more helpful than the similarity based on the co-occurrences of words/terms in threats because of the problems mentioned in Section IV-A. Each threat in ST is mitigated/avoided by the set of security functions, i.e., SFRs. SFR-commonality focuses on the co-existences of means for such mitigation. There are generally several alternative means to satisfy a requirement, and a threat, which is a kind of attacker's requirement, is also satisfied in the same way. When attackers want to threaten assets, they at least utilise parts of similar or the same vulnerabilities, which correspond to weaknesses of assets [23]. Because the limited means against such vulnerabilities, i.e., SFRs, are known at least in CC, using SFRs to identify similarities in our method seems to be a good rationale. This issue is empirically confirmed and explained in the next section.

We compare the Jaccard, Dice and Simpson indices to calculate appropriate commonality between two sets. The Dice and Simpson indices are defined in (7) and (8).

$$Dice = \frac{2 \times |X \cap Y|}{|X| + |Y|} \tag{7}$$

$$Simpson = \frac{|X \cap Y|}{min(|X|, |Y|)} \tag{8}$$

The Dice index takes almost the same value as the Jaccard index. The Simpson index is not suitable for pairs of sets, where the sizes of sets vary greatly. We thus used the Jaccard index in our method.

V. Evaluation

A. Preparation

We tested and confirmed that the method in Section IV-C worked well. We expect the method contributes to deriving threat pairs useful for expanding the candidates of security requirements. We thus focus on the recall and precision of EST. Because EST depends on a threshold for SFR-commonality and we subjectively determine the threshold by examining estimated gain, we especially confirm that estimated gain is a good clue to determine the threshold.

We use the following metrics in addition to the metrics in Section IV-B for our evaluation.

- *Recall*:
  Recall here refers to the degree to which how the decision by SFR-commonality can find as many correct pairs as possible. Recall can be defined in (9).

  $$\frac{|COR \cap EST|}{|COR|} \tag{9}$$

  In (9), $|S|$ is the number of elements in a set, S. The recall is about 0.66 (=2/3) in Figure 3. We expect recall will become 1.0 as much as possible because we want to find as many candidates for security requirements as possible.
- *Precision*:
  Precision here refers to the degree to which the decision by SFR-commonality is precise. Precision can be defined in (10).

  $$\frac{|COR \cap EST|}{|EST|} \tag{10}$$

TABLE I. STS IN OS DOMAIN

ID	No.# of Threats	Summary
MacOS	3	Apple Mac OS 10.6, Dec. 2009
RedHat	7	Red Hat Enterprise Linux Ver. 5.3 for CAPP Compliance on Dell 11th Generation PowerEdge Servers, Dec. 2009
Linux	13	Wind River Linux Secure 1.0, Apr. 2011

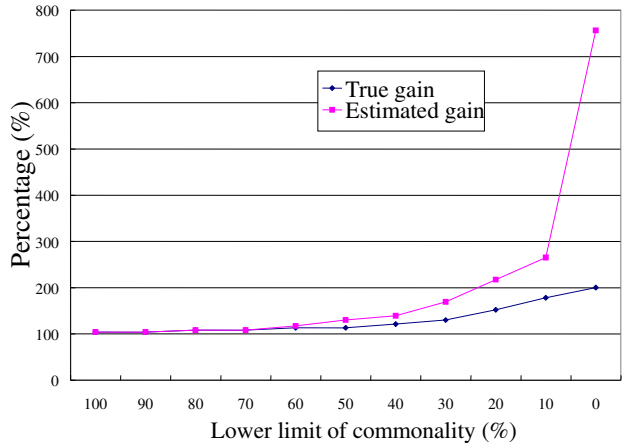


Figure 5. Changes in true and estimated gain of threat pairs in OS domain.

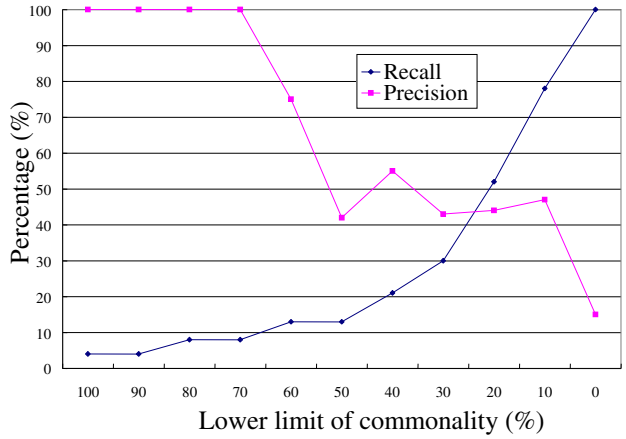


Figure 6. Changes in recall and precision of threat pairs in OS domain.

The precision in Figure 3 is 0.4 (=2/5). We also expect precision will become 1.0 as much as possible because we do not want to investigate threats that are not necessary. However, we can accept a certain degree of imprecision because precision and recall have trade-offs.

- *True gain*:
  Apparently, not all estimated pairs give us useful suggestion for finding additional candidates for security requirements because not all are correct pairs. We thus want to know the actual gain caused by estimated pairs if we can identify the correct pairs. The true gain indicates such a degree, and is defined in (11).

$$\frac{|EST \cap COR| + |ALL|}{|ALL|} \quad (11)$$

The true gain in Figure 3 is about 1.3 (=(2+7)/7). We, of course, expect the true gain to be sufficiently large because missing threats to be investigated are avoided.

Because COR is necessary for evaluation, one author and his student decided it. They checked all potential threat pairs whether a threat and another in each pair respectively threatened the same type of functional requirements.

B. Results

We observed changes in the gains in operating systems (OSs) according to the changes in the lower limit of SFR commonality in Figure 5. The STs are summarised in Table I. Because there were a total of 23 threats and there were 23 correct pairs, the upper limit of true gain is 200 % as shown in the figure. Because there were a total 23 threats and there were 151 potential pairs, the upper limit of estimated gain was about 750 % (100*(23+151)/23) as well. As we can see from Figure 5, estimated gain largely increased when the limit changed from 10 to 0 %. According to our method presented in Section IV-C, 10 % was a suitable lower limit for SFR-commonality, i.e., the threshold. Because the precision and recall in Figure 6 were about 45% and 80% at the 10 % lower limit, estimated gain seemed to be a good clue to determine a suitable threshold.

We next used three STs in IC chip domain. Three are summarized in Table II. We used STs written in either English or Japanese in this domain. Because SFR-commonality is independent of the language used in individual STs, we could use such STs together. There were number 174 potential pairs, and 15 correct pairs. We plotted the changes in true and estimated gain of threat pairs in Figure 7 in the same way as that in Figure 5. The most suitable lower limit also seemed to be 10 % in this domain. We then plotted the changes in recall and precision in Figure 8 in the same way as that in Figure 6. Recall became sufficiently accurate (80%) at this lower limit according to this figure, while precision was acceptable (about 22%). The results indicated that most correct pairs could be estimated, and the errors in the estimates were acceptable. We also regarded our method as working well in this IC domain.

We finally used three STs in the domain of multi-function devices (MFDs). MFDs are also called multi-function printers (MFPs). The three STs are summarised in Table III. We used STs written in Japanese in this domain. There were 24 potential pairs, and four correct pairs. According to the changes in estimated gain in Figure 9, the most suitable lower limit was 20 or 10 %. As we can see from in Figure 10, recall had been about 75% from the 100 to 10 % lower limit. One of the reasons for this was that there were not that many correct pairs in this domain. This was one of the reasons recall has been good. Precision decreased from the 10 % lower limit, as indicated in Figure 10. These results in Figure 10 indicated that 20 % lower limit was the best one. This could

 ISBN: 978-1-61208-432-9

TABLE II. STS IN IC CARD DOMAIN.

ID	No.# of Threats	Summary
C191	6	IC card for residents, Oct. 2008. in Japanese
C210	9	Farmware for Mobile FeliCa IC chip, Feb. 2009, in Japanese
C229	8	Apollo OS e-Passport V1.0, Jul. 2009, in English

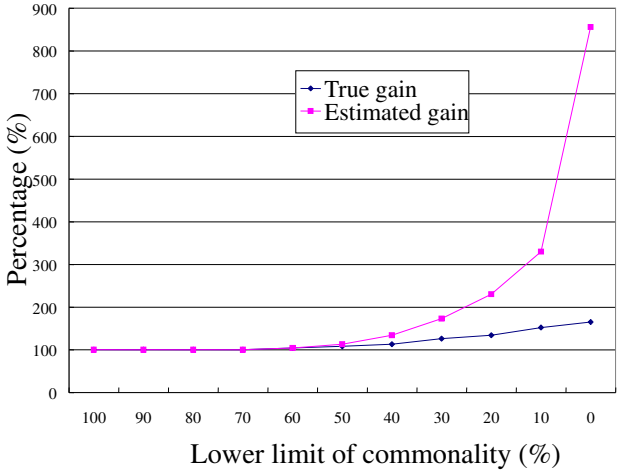


Figure 7. Changes in true and estimated gain of threat pairs in IC domain.

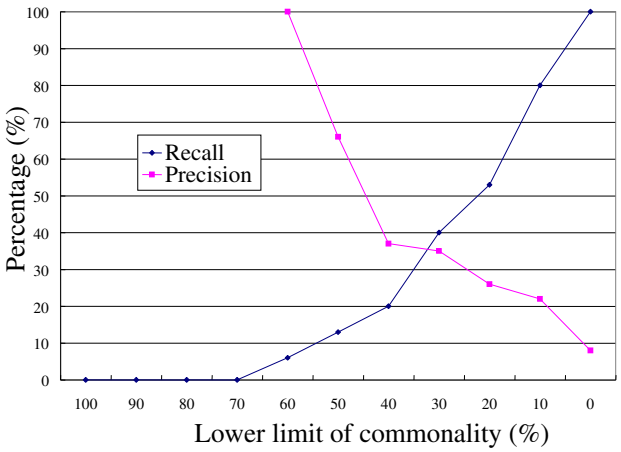


Figure 8. Changes in recall and precision of threat pairs in IC domain.

be estimated from the change in estimated gain in Figure 9 explained above. We also regarded our method worked well in this MFD domain.

C. Discussion

The method seemed to work well because the threshold could be decided by observing the changes of estimated gain. For each domain in this evaluation, the estimated gain suddenly and largely increased when the lower limit of commonality was nearby zero. We regarded the lower limit of commonality just before largely increasing as the threshold. At the threshold, the recall was more than 70% for each domain while the precision was 20 to 40 %. Because the goal of this method is to find potential threats as much and efficiently as possible, we may regard the results were acceptable.

We will now discuss the threats to validity in the evaluation. Most metrics in the evaluation depended on both estimated and correct pairs. Because the estimated pairs and correct pairs were defined separately, we did not worry about threats to internal validity. We used three different application domains for our evaluation, and the STs in each domain were written in two different languages, only in English, in English and Japanese, and only in Japanese. We thus regarded we had taken care of the problems caused by external validity. When the total number of STs is not very large, a requirements analyst does not have to use our method but simply investigates all the threats. Because it takes considerable effort to find correct pairs, only three STs were integrated in each domain and only 10 to 20 threats were included in the evaluation. There is no upper limit of the number on STs when using our method in actual situations because we do not have to find correct pairs and we can systematically calculate estimated pairs and gain as was explained in Section IV-C. We can thus derive a huge number of threat pairs where no analyst can investigate all the threats. The metrics in Section IV-B and Section V-A seem to be reasonable for measuring what we want to know, and most of them are systematically derived from the ST documents. However, only correct pairs should be subjectively determined. The correct pairs in our evaluation are determined at least two researchers so that we decreased threats to construct validity. Transitivity of pairs is not applied when true and estimated gains were calculated, which was also explained in Section IV-B. Therefore, actual gains would have been underestimated in our evaluation. However, these underestimates did not have adverse effects on the results we obtained from our evaluation because estimated gain was only used for predicting the appropriate threshold of commonality. Because we did not apply statistical tests, threats to conclusion validity still remained.

VI. CONCLUSION

We proposed and evaluated a method of deriving threat pairs from several STs for security requirements elicitation. One threat in an ST in the method is related to another threat in another ST according to SFR-commonality, which is calculated on the basis of SFRs used in both threats. Several STs are thus integrated together (response to RQ1). Threat pairs contribute to expanding the candidates for security requirements when a security requirement against a threat in an ST has already been found. Because SFR does not depend on languages such as English or French, we can integrate STs written in any languages (response to RQ2). We need a threshold to determine whether two threats with a certain SFR-commonality may really be examined together as candidates for security requirements. The threshold can be decided by observing the change of estimated gain, which could be calculated without knowledge as to whether two threats really threaten the same requirement respectively (response to RQ3).

 ISBN: 978-1-61208-432-9

TABLE III. STS IN MULTI-FUNCTION DEVICE (MFD) DOMAIN.

ID	No.# of Threats	Summary
C291	2	Toshiba Tec, e-Studio 555/655/755/855, Jun. 2009, in Japanese
C272	2	Kyocera, Data Security Kit (E) Software Type IV ver. 1.10, Aug. 2010, in Japanese
C281	5	Sharp, MX-FR22 ver. 0.05, Jul. 2011, in Japanese

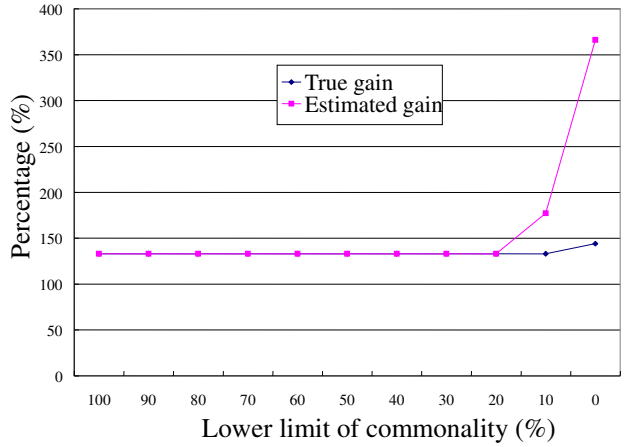


Figure 9. Changes in true and estimated gain of threat pairs in MFD domain.

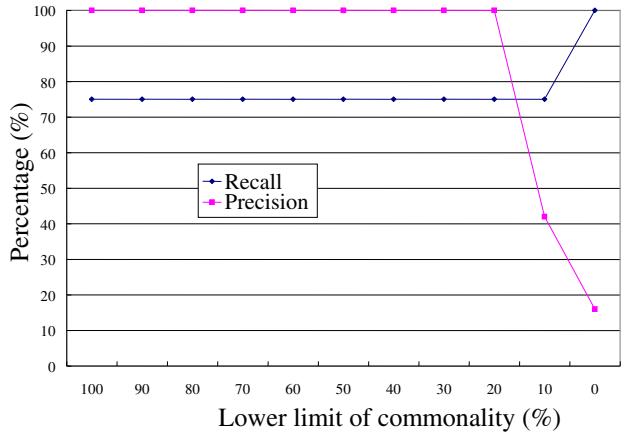


Figure 10. Changes in recall and precision of threat pairs in MFD domain.

There are a great deal of security knowledge in the field of security [18], and some of it is highly structured, is easy for computers to handle, and is usually up to date. For example, Common Attack Patterns Enumeration and Classification (CAPEC) [8] provides knowledge on attacks with several useful attributes such as attack prerequisites (preconditions of an attack) and typical ways of mitigation although this knowledge is mainly about design or implementation issues. Knowledge is also provided in XML documents. We want to extend our method so that it can be applied to various kinds of structured and machine-readable security documents.

ACKNOWLEDGMENT

This work was supported by KAKENHI 15H02686.

REFERENCES

[1] K. Breitman and J. C. S. do Prado Leite, “Ontology as a requirements engineering product,” in RE, 2003, pp. 309–319.

[2] S. W. Lee and R. A. Gandhi, “Ontology-based active requirements engineering framework,” in APSEC, 2005, pp. 481–490.

[3] H. Kaiya and M. Saeki, “Using domain ontology as domain knowledge for requirements elicitation,” in RE, 2006, pp. 186–195.

[4] D. V. Dzung and A. Ohnishi, “Improvement of quality of software requirements with requirements ontology,” in QSIC, 2009, pp. 284–289.

[5] M. Kitamura, R. Hasegawa, H. Kaiya, and M. Saeki, “An integrated tool for supporting ontology driven requirements elicitation,” in ICSOFT (SE), 2007, pp. 73–80.

[6] I. Omoronyia, G. Sindre, T. Stålhane, S. Biffl, T. Moser, and W. D. Sunindyo, “A domain ontology building process for guiding requirements elicitation,” in REFSQ, 2010, pp. 188–202.

[7] “Certified Products : New CC Portal,” http://www.commoncriteriaportal.org/products/ [accessed: 2015-08-19].

[8] “CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC),” http://capec.mitre.org/ [accessed: 2015-08-19].

[9] M. Saeki, S. Hayashi, and H. Kaiya, “Enhancing Goal-Oriented Security Requirements Analysis Using Common Criteria-Based Knowledge,” International Journal of Software Engineering and Knowledge Engineering, vol. 23, no. 5, Jun. 2013, pp. 695–720.

[10] Y. Zhao, J. Dong, and T. Peng, “Ontology classification for semantic-web-based software engineering,” IEEE T. Services Computing, vol. 2, no. 4, 2009, pp. 303–317.

[11] F. Massacci, J. Mylopoulos, F. Paci, T. T. Tun, and Y. Yu, “An extended ontology for security requirements,” in Advanced Information Systems Engineering Workshops - CAiSE 2011 International Workshops, London, UK, June 20-24, 2011. Proceedings, 2011, pp. 622–636.

[12] A. Souag, C. Salinesi, R. Mazo, and I. Comyn-Wattiau, “A security ontology for security requirements elicitation,” in Engineering Secure Software and Systems - 7th International Symposium, ESSoS 2015, Milan, Italy, March 4-6, 2015. Proceedings, 2015, pp. 157–177.

[13] J. S. Dong, Y. Feng, Y. F. Li, and J. Sun, “A tools environment for developing and reasoning about ontologies,” in APSEC, 2005, pp. 465–472.

[14] A. Zouaq and R. Nkambou, “Evaluating the generation of domain ontologies in the knowledge puzzle project,” IEEE Trans. Knowl. Data Eng., vol. 21, no. 11, 2009, pp. 1559–1572.

[15] M. Li and F. Zang, “A self-feedback methodology of domain ontology modeling,” Software Engineering, World Congress on, vol. 2, 2009, pp. 218–223.

[16] K. K. Breitman and J. C. S. do Prado Leite, “Lexicon based ontology construction,” in SELMAS, 2003, pp. 19–34.

[17] M. Karyda, T. Balopoulos, L. Gymnopoulos, S. Kokolakis, C. Lambrinoudakis, S. Gritzalis, and S. Dritsas, “An ontology for secure e-government applications,” in Proceedings of the The First International Conference on Availability, Reliability and Security, ARES 2006, The International Dependability Conference - Bridging Theory and Practice, April 20-22 2006, Vienna University of Technology, Austria, 2006, pp. 1033–1037.

[18] A. Hazeyama, “Survey on body of knowledge regarding software security,” in SNPD, 2012, pp. 536–541.

[19] M. Saeki and H. Kaiya, “Security requirements elicitation using method weaving and common criteria,” in MoDELS Workshops, 2008, pp. 185–196.

[20] K. Taguchi, N. Yoshioka, T. Tobita, and H. Kaneko, “Aligning security requirements and security assurance using the common criteria,” in

Proc. 4th International Conference on Secure Software Integration and Reliability Improvement (SSIRI'10), 2010, pp. 69–77.

[21] M. Ware, J. Bowles, and C. Eastman, "Using the common criteria to elicit security requirements with use cases," in Proc. IEEE Southeast Conference, 2005, pp. 273–278.

[22] P. Jaccard, "The distribution of the flora in the alpine zone," New Phytologist, vol. 11, 1912, pp. 37–50, doi: 10.1111/j.1469-8137.1912.tb05611.x.

[23] International Standard, " ISO/IEC 27002 Information technology - Security techniques - Code of practice for information security management," Jun. 2005.

 ISBN: 978-1-61208-432-9