The Journal of Systems and Software 165 (2020) 110566

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

An ontology-based learning approach for automatically classifying security requirements

Tong Li*, Zhishuai Chen

Faculty of Information Technology, Beijing University of Technology, Beijing, China

ARTICLE INFO

Article history:
Received 31 May 2018
Revised 4 January 2020
Accepted 28 February 2020
Available online 29 February 2020

Keywords:
security requirements ontology
security requirements classification
linguistic pattern
natural language processing
machine learning

ABSTRACT

Although academia has recognized the importance of explicitly specifying security requirements in early stages of system developments for years, in reality, many projects mix security requirements with other types of requirements. Thus, there is a strong need for precisely and efficiently classifying such security requirements from other requirements in requirement specifications. Existing studies leverage lexical evidence to build probabilistic classifiers, which are domain-dependent by design and cannot effectively classify security requirements from different application domains. In this paper, we propose an ontology-driven learning approach to automatically classify security requirements. Our approach consists of a conceptual layer and a linguistic layer, which understands security requirements based on not only lexical evidence but also conceptual domain knowledge. In particular, we apply a systematic approach to identify linguistic features of security requirements based on an extended security requirements ontology and linguistic knowledge, connecting the conceptual layer with the linguistic layer. Such linguistic features are then used to train domain-independent security requirements classifiers by using machine learning techniques. We have carried out a series of experiments to evaluate the performance and generalization ability of our proposal against existing approaches. The results of the experiments show that the proposed approach outperforms existing approaches with a significant increase of F_1 score (0.63 *VS.* 0.44) when the training dataset and the testing dataset come from different application domains, i.e., the classifiers trained by our approach can be generalized to classify security requirements from different domains.

1. Introduction

Security breaches have repeatedly caused millions of dollar losses per year to large organizations, and this cost is on the rise (Institute, 2018). Related research has shown that early security requirements analysis can reduce costs related to software development and maintenance from 12–21% (Hoo et al., 2001). Different from other non-functional requirements, security requirements are typically operationalized as functional requirements that can influence the design of system architecture. Moreover, such security requirements operationalization analysis requires particular security expertise. As such, security requirements have to be specially processed independently from other requirements. Although academia has recognized the importance of explicitly specifying security requirements in early stages of system developments for more than a decade, due to the lack of security expertise, in many projects security requirements are not explicitly identified and specified during requirements analysis. Instead, security requirements may implicitly spread throughout different parts of a requirements specification (Knauss et al., 2011).

Manually identifying which requirements are security requirements from an entire requirements specification is time-consuming and error-prone, arousing the need for automatic analysis. However, there are two main challenges to the automatic identification of security requirements. Firstly, there is no commonly accepted ontological definition for security requirements, and different people may interpret security requirements differently. As surveyed by Souag et al., there are eight security requirements ontologies have been proposed, each of which involves a particular combination of different security requirements concepts and targets a specific level of abstraction (Souag et al., 2012). For example, some ontologies include security objectives (Massacci et al., 2011; Kim et al., 2005), some take into account organizational issues (Fenz and Ekelhart, 2009; Mouratidis et al., 2003), and some define vulnerability as an imperative concept of security requirements (Elahi et al., 2009; Tsoumas and Gritzalis, 2006). Secondly, the inherent ambiguity of natural language makes the automatic identification of security requirements even more difficult. Especially, different people may use different syntax and terms to describe security requirements. Although recent research has

* Corresponding author.
E-mail address: litong@bjut.edu.cn (T. Li).

https://doi.org/10.1016/j.jss.2020.110566
0164-1212/© 2020 Elsevier Inc. All rights reserved.

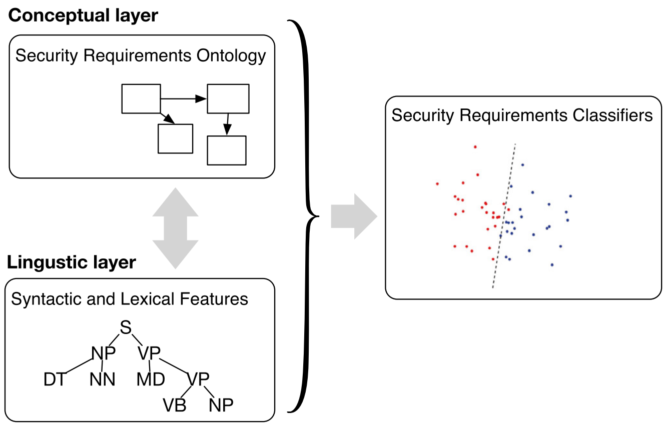

Conceptual layer
Security Requirements Ontology
Lingustic layer
Syntactic and Lexical Features
S
NP
VP
DT
NN
MD
VP
VB
NP
Security Requirements Classifiers

Fig. 1. The essential idea of our proposal

introduced specific templates for documenting security requirements (Rudolph et al., 2016), in practice, many companies specify security requirements based on their own templates.

Existing solutions to the automatic identification of security requirements rely on supervised machine learning techniques, i.e., training security requirements classifiers. Cleland-Huang et al. mine indicators from requirements specification based on their frequency, which are then used to train non-functional requirements classifiers (Cleland-Huang et al., 2007). Knauss et al. use statistical methods to train security requirements classifiers, considering all words appear in a requirement instead of only keywords (Knauss et al., 2011). Such approaches classify security requirements based on statistical evidence at the lexical level (e.g., term frequency), but do not really “comprehend” the meaning of requirements, as no syntactic and semantic analysis was performed. As a result, the classifiers trained by their approaches are likely to overfit security requirements in particular application domains. Thus, when such classifiers are used to analyze specifications from a new application domain, which were not included in the training dataset, the results are normally not acceptable.

We argue that successful recognition of security requirements concepts and their interrelationships will contribute to the identification of security requirements. To this end, our previous work has defined a set of linguistic rules of security requirements based on security requirements and linguistic knowledge in order for the recognition of security concepts (Li, 2017). Each rule indicates a particular representation of textual security requirements, which can be automatically matched with textual requirements using a proposed syntactic- and keyword-based method. Subsequently, we have trained domain-independent security requirements classifiers based on such rules.

In this paper, we extend our previous work and propose an ontology-based learning approach to systematically deal with the automatic classification of security requirements. Specifically, our extensions contribute to the following aspects:

1. We define a formal security requirements ontology at the conceptual layer based on an in-depth investigation of existing security requirements ontologies (Section 2). Specifically, we adopt Description Logic (DL) to specify a logical formalism for the proposed ontology, rendering precise and unambiguous definitions of security requirements.
2. In order to relate the ontology at the conceptual layer to syntactic and lexical features of textual security requirements at the linguistic layer (as shown in Fig. 1), we previously proposed a preliminary list of security keywords for identification of security requirements concepts by manually examining existing security taxonomies. This paper proposes a systematic linguistic analysis process to automatically mine security keywords of core security requirements concepts (e.g., threat and security control). In particular, by going through the systematic process, we have significantly enriched security keywords that are used to identify security concepts (Section 3.2)
3. We have evaluated our extended proposal based on the same dataset we used in our previous work, the results of which show that our extended proposal can improve the performance of our security requirements classifier by 1-2% F_1 score (Experiments 2-3 in Section 5.5). In addition, we further evaluate our proposal based on another security requirements dataset, examining the performance of our approach on security requirements of different qualities. The results show that our approach works significantly better on security requirements that are well specified (Experiments 4 in Section 5.5).
4. We have extensively surveyed and analyzed related studies in the past decade (Section 7), and have carried out a series of in-depth discussion regarding the applicability and open challenges of our approach (Section 6).

In the remaining part of this paper, we first present the extended security requirements ontology in Section 2, establishing the conceptual layer of our approach. In Section 3, we define linguistic features of security requirements based on the ontology, associating the conceptual layer with the linguistic layer. After that, we describe details of identifying linguistic features from textual requirements specification and training security requirements classifiers in Section 4. We evaluate the obtained security requirements classifiers on three industry requirements specifications and 15 publicly available datasets from PROMISE Software Engineering Repository, the results of which are presented in Section 5 and discussed in Section 6. Section 7 presents existing approaches and compares them with our proposal. Finally, we conclude the paper and discuss future work in Section 8.

2. A comprehensive security requirements ontology

In this section, we define a comprehensive security requirements ontology, which involves various views of security requirements, based on existing security requirements ontologies and taxonomies. The rationale of investigation security requirements ontology is to explicitly represent possible ways of defining security requirements. However, there is no widely accepted definition for security requirements. For instance, security requirements have traditionally been considered to be non-functional requirements that do not have clear-cut satisfaction criteria (e.g., the system should provide confidentiality); while other researchers argue security requirements are constraints put on particular system functions (Haley et al., 2008).

As surveyed by Souag et al., more than 20 security requirements-related ontologies have been proposed by different researchers (Souag et al., 2012). Although most of such ontologies focus on security ontologies in a general sense and are not specialized in security requirements, the different concepts defined in such ontologies have indicated various concerns of security requirements. In particular, an *asset* is anything that has value to an organization (e.g., personal data), which is typically concerned by security requirements. A *security property* specifies a characteristic of security (e.g., confidentiality, integrity, and availability), which indicates the security objective that a security requirement aims to achieve. A *threat* presents an undesired state that an attacker wants to impose on the targeting system (e.g., threat of theft), which harms assets and thus damages corresponding security properties. A *countermeasure* is a solution

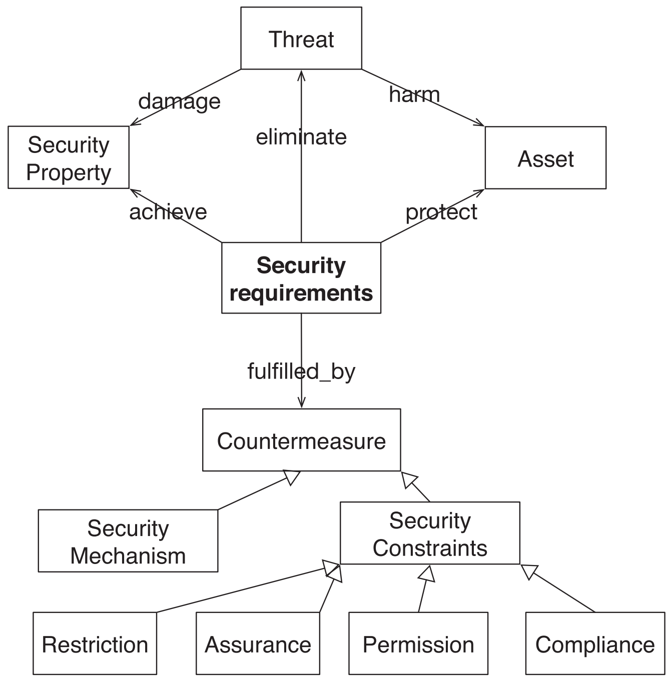


Fig. 2. A revised conceptual model for security requirements

that is adopted to fulfill corresponding security requirements. In theory, a threat may result in an attack, i.e., they are two different concepts. For example, the threat of theft may result in an actual theft (attack). However, as pointed by Tulloch, in some books and articles, "attack" and "threat" are confounded by being used as synonyms (Tulloch, 2003). As such, in this paper, we do not distinguish these two concepts.

Based on the surveyed concepts, we define a comprehensive security requirements ontology in which *security requirements* are defined as an essential concept that is connected with other concepts through particular relations (Fig. 2). Specifically, we argue that each concept that is linked to security requirements indicates a particular view of security requirements, implying a plausible way of defining security requirements. For instance, a security requirement can be defined in terms of the elimination of threats, e.g., "a system shall resist denial of service attacks". For another example, stakeholders may express their security requirements by emphasizing the assets they want to protect, e.g., "system servers should be protected from attacks".

As pointed out by Firesmith in Firesmith (2003), most requirements engineers are poorly trained to specify security requirements and often confuse them with security countermeasures. As a result, many practitioners write security requirements in terms of particular security countermeasures that are to be implemented. As an inclusive ontology, we accept such a definition of security requirements in this paper. Furthermore, we elaborate on the concept of security countermeasure into two sub-classes of countermeasures for better distinction. A *security mechanism* is a function that is designed to detect, prevent, or recover from a security attack, and thus fulfill one or more security requirements. A *security constraint* is a constraint that is imposed on existing system functions in order to protect assets from harm. In particular, we have categorized security constraints into four sub-classes that have been practically used by researchers to represent security constraints: *restriction, assurance, permission*, and *compliance*. Each of these four types of security constraints is presented in a particular syntactic structure, which will be detailed in the next subsection.

In order to provide unambiguous definitions to security requirement concepts and relations that are used in our approach, we have formally defined the security requirements ontology by using Description Logic (DL) (Baader, 2003). Specifically, the reasons of choosing DL fall in the following aspects:

1. DL is easy to balance between expressivity and reasoning complexity, as it supports different sets of mathematical constructors, e.g., cardinality restrictions and role hierarchy. In particular, it is mainly used for classifying concepts and instances via reasoning. For example, As shown in Table 1, Axiom 1 defines a security requirement as a requirement that either achieves a security property, or eliminates a threat, or protect an asset or fulfilled by a countermeasure. This precisely defines our rationale of classifying a security requirement. For another example, as specified in Axiom 2, the concept hierarchy defined by DL indicates that security constraint is a particular type of countermeasures, contributing to the recognition of security requirements concepts.
2. The Web Ontology Language (OWL) is one of the most popular ontological language, many existing security ontologies have been defined in terms of OWL. OWLâs profile is based on DL, and thus can be easily transformed into DL. As such, it is possible for us to reuse existing ontology knowledge that has been defined in OWL or DL.
3. DL-based reasoning can be complemented by Semantic Web Rule Language (SWRL) when necessary, allowing for more sophisticated semantic analysis in future.

Table 1 lists a few definitions of our security requirements ontology that are relevant to the classification of security requirements. In particular, as shown in Fig. 2, each concept that is directly associated with the concept of *Security Requirements* is typically used as a particular way for describing security requirements. Such ontological definitions serve as the basis of defining linguistic features of security requirements, which will be detailed in Section 3.

3. Identifying linguistic features of security Requirements

To efficiently classify security requirements, we need to train classifiers that understand the semantics of security requirements. Thus, we propose to establish linkages from the semantics of security requirements to their textual representations, i.e., associating security requirements ontology with linguistic features of security requirements. With the help of such linkages, we are able to automatically recognize security requirements concepts via text processing, contributing to the identification of security requirements. In this section, we propose a systematic process for obtaining linguistic features of textual security requirements based on the ontology defined in the last section. In particular, the identified linguistic features consist of security requirements linguistic rules and keywords, which are detailed respectively in the remainder of this section.

3.1. Defining linguistic rules of security requirements

Based on the ontological definition of security requirements described in Table 1, we define a collection of linguistic rules as linguistic features of security requirements (Tables 2 and 3). In particular, we first consider atomic rules, i.e., defining security requirements based on an individual security requirements concept. On top of such atomic rules, we further consider their combination, yielding compound rules. We argue that requirements that can be matched with one or multiple proposed linguistic rules are much more likely to be security requirements than those that cannot be matched. These two types of rules are detailed below.

Table 1
Ontological definitions of security requirements

No.	Definitions
1	$SecurityRequirement \equiv Requirement \sqcup \exists achieve.SecurityProperty \sqcup \exists eliminate.Threat \sqcup \exists protect.Asset \sqcup \exists fulfilled\_by.Countermeasure$
2	$Countermeasure \equiv SecurityMechanism \sqcup SecurityConstraints$
3	$SecurityConstraint \equiv Restriction \sqcup Assurance \sqcup Permission \sqcup Compliance$

Table 2
Atomic linguistic rules of security requirements

Name	Rules
Threat-based (T)	$< Subject > \ < Eliminate > \ < Threat >$
Asset-based (A)	$< Subject > \ < Protect > \ < Asset >$
Security property-based (S)	$< Subject > \ < Achieve > \ < SecurityProperty >$
Countermeasure-based (C)	$< Subject > \ < Provide > \ < SecurityMechanism >$
	$< Subject > \ < Restrict > [< Object >]$
	$< Subject > \ < Assure > [< SecureCondition >]$
	$< Subject > \ < Permit > [< SecureCondition >]$
	$< Subject > \ < Obey > [< SecurityPolicy >]$

Table 3
Compound linguistic rules of security requirements

Name	Rules
CA	$< Subject > \ < Provide > \ < SecurityMechanism > to < Protect > \ < Asset >$
CT	$< Subject > \ < Provide > \ < SecurityMechanism > to < Eliminate > \ < Threat >$
CS	$< Subject > \ < Provide > \ < SecurityMechanism > to < Achieve > \ < SecurityProperty >$
TA	$< Subject > \ < Eliminate > \ < Threat > to < Protect > \ < Asset >$
SA	$< Subject > \ < Protect > \ < SecurityProperty > of < Asset >$
TS	$< Subject > \ < Eliminate > \ < Threat > to < Achieve > \ < SecurityProperty >$
CAT	$< Subject > \ < Provide > \ < SecurityMechanism > to < Protect > \ < Asset > from < Threat >$
CTS	$< Subject > \ < Provide > \ < SecurityMechanism > to < Eliminate > \ < Threat > to < Achieve > \ < SecurityProperty >$
CSA	$< Subject > \ < Provide > \ < SecurityMechanism > to < Protect > \ < SecurityProperty > of < Asset >$
ATS	$< Subject > \ < Protect > \ < Asset > from < Threat > to < Achieve > \ < SecurityProperty >$
CATS	$< Subject > \ < Provide > \ < SecurityMechanism > to < Protect > \ < Asset > from < Threat > to < Achieve > \ < SecurityProperty >$

Atomic Rules. As the first rule defined in Table 1, a security requirement can be alternatively defined in terms of security properties that are to be achieved, threats that are to be eliminated, assets that are to be protected and countermeasures that are to be implemented. Thus, in Table 2, we define atomic rules based on each of these definitions, respectively. For example, as security requirements should eliminate system threats, analysts may express security requirements based on system threats, resulting in the *threat-based rule* in Table 2. Specifically, this rule indicates sequential relations among the three security requirements concepts (i.e., terms that are within angle brackets), from left to right. It is worth noting that although the atomic rules listed in Table 2 are specified in active forms, their corresponding passive forms are also taken into account when matching such rules, which will be introduced in Section 4.

According to the bottom part in Fig. 2, *Countermeasure* can be specialized as *Security Mechanism* and *Security Constraint*, and the later one can further be specialized into four subclasses. We have defined particular rules for each of these subclassess, which are shown in the bottom row in Table 2. For example, the linguistic rule regarding security mechanism specifies a semantic structure that a *Subject Provides* a particular *Security Mechanism*. Note that concepts that are within brackets mean they are optional to match. Overall, we define 8 atomic rules based on the ontological definition shown in Table 1, i.e., a threat-based rule, an asset-based rule, a security property-based rule, and 5 countermeasure-based rules.

Compound Rules. The atomic rules shed light on the simplest ways of expressing security requirements, while compound rules focus on more complicated cases. Specifically, a compound rule involves multiple security requirements concepts, orchestrating them in an appropriate linguistic structure. In such a way, security requirements are expressed in a more comprehensive manner. By exploring all possible combinations among the 8 atomic rules, we have defined 35 compound rules, which are shown in Table 3. Specifically, the name of each compound rule consists of acronyms of security requirements concepts that are annotated in Table 2. For example, the last rule in Table 3 (i.e., CATS) combines all security requirements concepts, specifying countermeasures that need to provide, assets that need to protect, threats that need to eliminate, and security properties that need to achieve.

It should be noted that we show only one compound rule for each countermeasure-related rule in Table 3 in order to save space. Each of such countermeasure-related rules can actually be expanded to 5 rules according to the definitions of countermeasure-based rules in Table 2. For example, the first three concepts of the *CA* compound rule (i.e., the first row in Table 3) can be replaced by $< Subject > \ < Restrict > \ < Object >$, yielding another *CA* compound rule.

3.2. Mining security requirements keywords

Keywords have been used as indicators for classifying Non-Functional Requirements (NFRs) (Cleland-Huang et al., 2006), which shows promising results. Such NFR keywords are domain-general, and thus can be used to train classifiers that work well in different domains. Thus, apart from the above linguistic rules, we leverage security requirements keywords as another type of linguistic features of security requirements. If a requirement contains one or multiple such keywords, indicating the requirement may involve specific security requirement concepts and it is likely to be a security requirement.

Based on our proposed security requirements ontology, we define a security keyword as a word/phrase that represents one of the security concepts, i.e., *asset, security property, threat,*

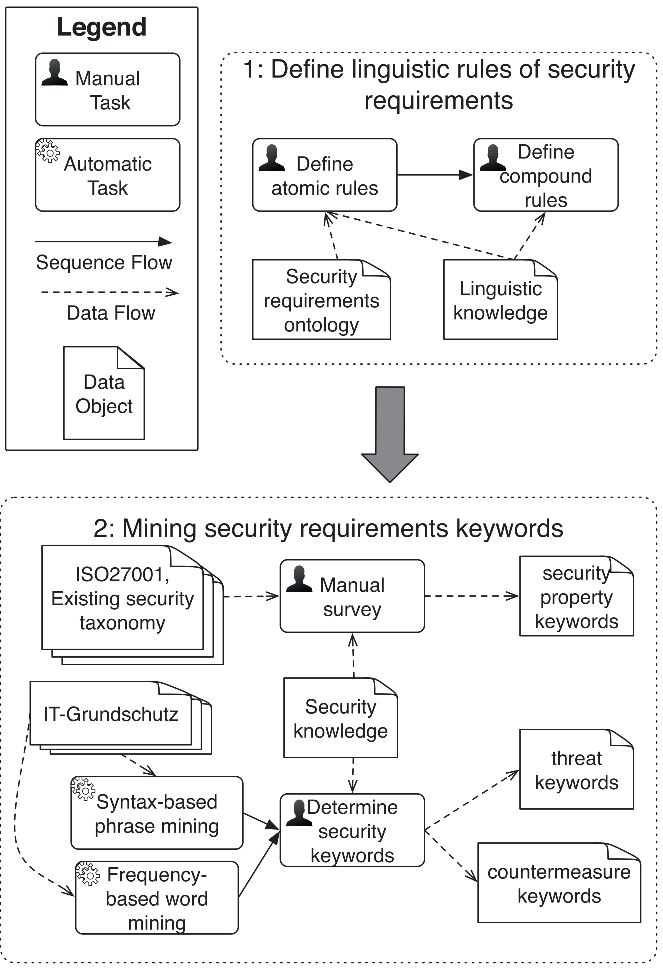

Legend
Manual Task
Automatic Task
Sequence Flow
Data Flow
Data Object
1: Define linguistic rules of security requirements
Define atomic rules
Define compound rules
Security requirements ontology
Linguistic knowledge
2: Mining security requirements keywords
ISO27001, Existing security taxonomy
Manual survey
security property keywords
IT-Grundschutz
Security knowledge
threat keywords
Syntax-based phrase mining
Determine security keywords
Frequency-based word mining
countermeasure keywords

Fig. 3. A systematic process for identifying linguistic features of security requirements

countermeasure. Note that as *asset* can be anything that is valuable to stakeholders, there are no particular keywords for describing assets. Thus, we focus on identifying keywords of the other three security requirements concepts. Since there are only a limited number of well-known security properties, we manually extract their corresponding keywords from existing security taxonomies (Firesmith, 2005; 2004; Souag et al., 2015) and security standards (Committee, 2012) based on our security expertise, as indicated in the top of Fig. 3 (part 2).

Different from security property, the concepts of threat and countermeasure involve much more security requirements keywords, requiring a careful check over comprehensive and professional corpora. Specifically, we have been aware of several candidates of such corpora, i.e., IT-Grundschutz (Committee, 2005a), NIST 800-53 (NIST, 2003), ISO27002 (Committee, 2005b), among which IT-Grundschutz is the most comprehensive one that contains detailed catalogues and descriptions of 380 threats and 927 security countermeasures (more than 2700 pages). Also, IT-Grundschutz is the only standard we have found that includes catalogues and descriptions for both threat and security countermeasure, while the other two only specify security countermeasures. As such, we eventually choose to use IT-Grundschutz as our security corpus.

We have designed and implemented a tool-supported method for automatically mining security requirements keywords for the aforementioned corpus, which combines frequency-based word mining and syntax-based phrase mining. The analysis process is

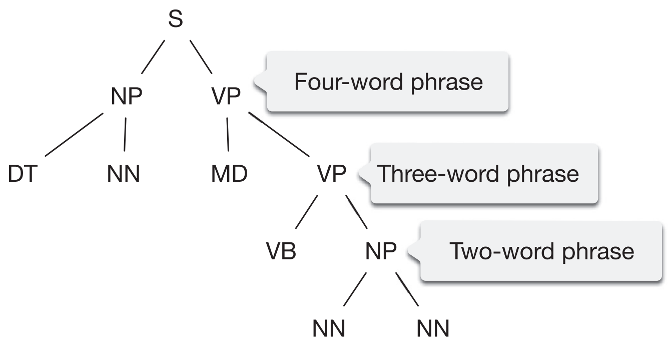


Fig. 4. Identification of potential phrases

shown at the bottom of Fig. 3. The technical details of these two mining techniques are detailed in the rest part of this subsection. After carrying out the automatic keyword mining process[1], we then manually inspect such results to obtain the eventual set of security requirements keywords, which are listed in Table 4. Specifically, as shown in the table we significantly enrich the keywords of *security mechanism* from 11 words to 45 words, and enrich the keywords of *threat* from 9 words to 50 words. With the increased number of keywords, our approach is able to identify more threats and security mechanisms, contributing to the training of security requirements classifiers. Section 5 will provide detailed evaluation results of this improvement.

Frequency-based word mining. To comprehensively mine keywords of threats and security mechanisms, we apply TF-IDF (Term FrequencyâInverse Document Frequency), which is a numerical statistic that can reflect how important a word is to a document in a collection of documents. Specifically, we treat each threat description as a document (d), and calculate *tf-idf* over all threat descriptions (D). Thus, for each word (w_i) in the document (d_j), we calculate its *tf-idf* according to Formula 1. In particular, term frequency $tf(w_i, d_j)$ shows how often w_i appears in d_j (Formula 2); while inverse document frequency $idf(w_i, D)$ measures how much information the word w_i provides, i.e., whether the term is common or rare across all documents. (Formula 3).

$$tf\text{-}idf(w_i, d_j, D) = tf(w_i, d_j) \times idf(w_i, D) \tag{1}$$

$$tf(w_i, d_j) = \frac{f(w_i, d_j)}{\sum_{w_k \in d_j} f(w_k, d_j)} \tag{2}$$

$$idf(w_i, D) = log\frac{|D|}{|d \in D : w_i \in d|} \tag{3}$$

After calculating *tf-idf* for all words in all threat documents, we rank them by documents and manually inspect the top 10 words of each threat document for their appropriateness of being threat keywords. Likewise, we mine security mechanism keywords in the same way.

Syntax-based phrase mining. In addition to mining single-word security keywords, we also aim to identify security phrases that consist of 2-4 words, such as “brute force”, “Trojan horse” etc. If a phrase repeatedly appears in the threat description documents, it is likely to be threat phrase. Thus, we first need to identify all potential phrases and then count their occurrences. Specifically, we parse each sentence in threat description documents in terms of abstract syntax tree, based on which we identify phrases that consist of 2-4 words. As illustrated in Fig. 4, within an abstract syntax tree, if a node can eventually be deducted to 2-4 leaf nodes, those

[1] All mining results are available online at https://www.dropbox.com/s/ruagh6bcxu8u8dh/jss.zip?dl=0.

Table 4
Mined keywords of security requirements concepts

Concept	Keywords
Security Property	security, identification, authentication, authorization, integrity, immunity, anonymity, confidentiality, privacy, availability, non-repudiation, accountability, business continuity, reliability
Security Mechanism	auditing, access control, proof, key, lock, pin, detection, validation, verification, awareness, education, training, back-up, clear desk, clear screen, clock synchronization, cryptography, disposal, log, monitoring, segregation, separation, password, screening, protection, compliance, restriction, assurance, permission, security management, digital signature, control, security gateways, firewall, encryption, security mechanism, safeguard, dmz, demilitarized zone, security check,security policy, security guideline, security setting, security strategy, security measure,security process, troubleshoot, diagnosis, key recovery, system recovery, trust network, certification, sandbox, antivirus, pki, ssl, puk, checklist, ipsec, virs scanning, awareness training, security training, adapted segmentation, galvanic separation, sealing, safety doors, automatic drainage, alarm system, uninterruptible power supply, ups, safeguard, safekeeping, anti-theft, surveillance
Threat	fraud, error, bug, vulnerability, risk, fault, leakage, misuse, virus, unauthorized asscess, unauthorized person, attacker, danger, error,trojan horse, brute force, damage, defect, burglary, flaw, violation, intrusion, steal, tamper, destruction, malware, hazard, leak, attack, manipulation, fraud, abuse, hijacking, spoof, falsify, detriment, abend, incorrect use, incorrect handling, incorrect entry, incorrect action, incorrect configuration, misjudgement, insecure connection, insecure channel, insecure policy, inadequate administration, improper, invalid, divulge
Asset	NP

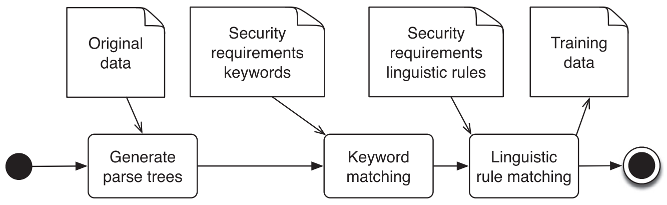
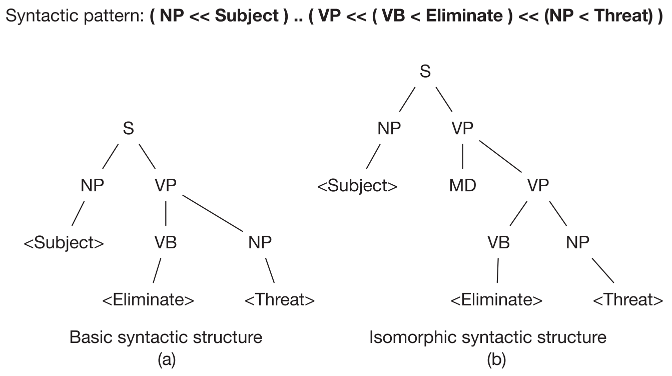


Fig. 5. The process matching the linguistic features


Syntactic pattern: (NP << Subject) .. (VP << (VB < Eliminate) << (NP < Threat))
S
NP
VP
<Subject>
VB
NP
<Eliminate>
<Threat>
Basic syntactic structure
(a)
S
NP
VP
<Subject>
MD
VP
VB
NP
<Eliminate>
<Threat>
Isomorphic syntactic structure
(b)
Fig. 6. The syntactic structure of the threat-based linguistic rule

leaf nodes are deemed as a potential threat phrase. After obtaining a list of potential threat phrases, we rank them based on their occurrences, and manually inspect such phrases to pick up threat phrases. Similarly, we perform the same analysis on security mechanism descriptions to discover corresponding keywords.

Eventually, we merge security requirements keywords mined from the aforementioned analysis, resulting in a comprehensive list of domain-independent keywords. In particular, there are 14 security property keywords, 57 threat keywords, and 69 security mechanism keywords, which are presented in Table 4. For one thing, such keywords are mined as linguistic features of security requirements for training security requirements classifiers; for another, they are also used for matching the proposed linguistic rules which are detailed in the next section.

4. Training security requirements classifiers

The proposed linguistic features are defined based on both security and linguistic knowledge, which reflect the essence of security requirements and thus are ideal features for training security requirements classifiers. In this section, we first present how to automatically match textual requirements specifications with those linguistic features, i.e., generating the training dataset. Then, we detail the settings of training security requirements classifiers.

4.1. Text processing for generating training dataset

The objective of text processing is to identify linguistic features in terms of security requirements keywords and linguistic rules. The process of matching the linguistic features consists of three steps: generate parse trees, keyword matching and linguistic rule matching (Fig. 5), which are detailed below.

4.1.1. Pre-processing

We first analyze the syntactic structure of each requirement by generating its corresponding abstract syntax tree, contributing to linguistic rules matching. An abstract syntax tree is an ordered, rooted tree that describes how a sentence can be deduced from its root. For example, Fig. 6(a) shows an abstract syntax tree that describes the basic syntactic structure of the threat-based linguistic rule. In particular, *Subject* should be a *NP (Noun Phrase)*, followed by which is a *VP (Verb Phrase)*. Moreover, the *VP* consists of a *VB (Verb)* which can deduce *Eliminate* and a *NP* which can deduce *Threat*. To generate such trees, we leverage Stanford Parser[2], which is a Java implementation of a lexicalized (Probabilistic Context-Free Grammar) PCFG parser. Next, to facilitate keyword matching, we need to ensure that a word in different inflected forms can be recognized as the same word. To this end, for each leaf node in the syntax tree, we reduce inflected (or sometimes derived) words to their word stem, i.e., stemming. For example, *validation, validate*, or *validated* will be stemmed as *valid* for matching.

4.1.2. Matching security requirements keywords

We match each stemmed leaf node in the parse tree with the predefined security requirements keywords, which are shown in Table 4. Once a keyword is found in a requirement, we then replace the matched word with its corresponding security requirements concept, contributing to matching linguistic rules.

[2] https://nlp.stanford.edu/software/lex-parser.html.

In addition to matching the essential security concepts, we also need to identify the conceptual relations among security requirements concepts (e.g., eliminate, protect, restrict in Table 2), which is another precondition for matching linguistic rules. Similar to the recognition of security requirements concepts, we match the conceptual relations with a list of synonyms based on WordNet Miller (1995) Specifically, for each conceptual relation, we retrieve all words from all its synsets and take them into account during the relation matching.

4.1.3. Matching linguistic rules

Each aforementioned linguistic rule (Tables 2 and 3) specifies a particular order among different security requirements concepts, showing an ideal representation of a security requirement. However, due to the complexity and ambiguity of natural languages, it is common to document security requirements with additional words or phrases, which cannot be exactly matched with the proposed linguistic rules. In addition, sometimes people specify security requirements in a passive form instead of an active form, which should also be appropriately matched. As a result, we propose to further analyze the syntactic structure of each linguistic rule, based on which the rule matching can be done in a more flexible manner.

Once obtaining the basic syntactic structure of each linguistic rule, e.g, Fig. 6, we define a corresponding syntactic pattern that characterizes a set of syntactic structures that are isomorphic to the basic one, e.g., the pattern that is shown on top of Fig. 6. Specifically, We use TGrep2 syntax[3] to specify such syntactic patterns, which can be matched automatically by using Tregex[4]. In particular, $A < < B$ means A is an ancestor of B, $A < B$ indicates A is the parent of B, and $A..B$ describes A precedes B (in depth-first traversal of tree). Note that $A < B < C$ means A is both the parent of B and the parent of C. According to such TGrep2 syntax, the syntactic pattern defined in on top of Fig. 6 can match an syntactic structure (i.e., Fig. 6(b)) that is isomorphic to the syntactic structure presented in Fig. 6(a). Specifically, the isomorphic syntactic structure contains an *NP* which is followed by a *VP*. The *NP* can eventually be deduced to *Subject*, and the *VP* can eventually be deduced to a *VB* and an *NP* which can be directly deduced to *Eliminate* and *Threat*, respectively.

4.2. Machine learning settings

As presented in Section 3, we take into account in total 35 linguistic rules and the 140 keywords as linguistic features of security requirements. All such features are defined as boolean features, i.e., we set the value of a feature as 1 if it is successfully matched in the textual requirements, otherwise we set it as 0.

According to the text processing method that we introduced in Section 4.1, we have developed a prototype tool to support the entire procedure. The tool implements various natural language processing features by using API provided by Stanford Parser, and can automatically determine values of the linguistic features over textual requirements. The original data to be processed is a list of requirements, each of which is an individual sentence (more details of the input data will be described in the next section), while the output is a featured dataset that will be used for training.

We leverage tools and APIs from Weka[5], which covers a broad spectrum of machine learning algorithms and can be easily integrated into our prototype tool. In particular, we first apply some data filters for data pre-processing, then make use of various types of classification algorithms to train security requirements classifiers, and finally use cross validation APIs for evaluation purpose.

Table 5
Statistics of the two datasets

Dataset	Specification	Req.	Sec. Req.	Pct.
Knauss et al. (2011)	ePurse	124	83	67%
	CPN	210	41	20%
	GP	176	63	36%
	Total	510	187	37%
Cleland-Huang et al. (2006)	Project_1	28	1	4%
	Project_2	39	3	8%
	Project_3	79	10	13%
	Project_4	54	10	19%
	Project_5	72	7	10%
	Project_6	73	5	7%
	Project_7	22	2	9%
	Project_8	92	15	16%
	Project_9	23	0	0%
	Project_10	52	1	2%
	Project_11	12	3	25%
	Project_12	21	3	14%
	Project_13	18	2	11%
	Project_14	15	2	13%
	Project_15	11	2	18%
	Total	625	66	11%

5. Evaluating security requirements classifiers

In this section, we first briefly describe datasets that are used in our experiments. Then we present a group of research questions that we aim to address in this paper. According to such questions, we design a series of experiments and eventually report our experiment results.

5.1. Security requirements datasets

We focus on two requirements datasets that have been used in Knauss et al. (2011) and Cleland-Huang et al. (2006), respectively. In particular, the first dataset[6] consists of three industrial requirements documents: the Common Electronic Purse specification (ePurse), the Customer Premises Network specification (CPN), and the Global Platform specification (GP). As shown in Table 5, these three requirements specifications contain 124, 210, 176 pieces of requirements, respectively. Most of the individual requirements consist of only one sentence, several others contain two sentences. Knauss et al. have identified 83, 41, 63 pieces of security requirements out of the three requirements specifications (Knauss et al., 2011), respectively. There are several benefits of using this dataset. Firstly, the requirements specifications come from industry, successfully classifying which can well demonstrate the utility of our approach. Secondly, the three specifications involve three different application domains, allowing us to evaluate the generalization of classifiers that are trained based on particular domains. Thirdly, as far as we have been aware, several approaches have been evaluated based on this dataset (Knauss et al., 2011; Munaiah et al., 2017; Li, 2017), which can be compared with our approach.

The second dataset[7] consists of 15 requirements specifications that are developed as projects by MS students at DePaul University. Both the number of requirements and the number of security requirements vary dramatically by projects. As shown in Table 5, project_15 has only 11 requirements, project_9 even has no security requirements. Overall, the percentage of security

[3] https://web.stanford.edu/dept/linguistics/corpora/cas-tut-tgrep.html.
[4] https://nlp.stanford.edu/software/tregex.html.
[5] http://www.cs.waikato.ac.nz/ml/weka/.

[6] The dataset is available online at http://www.se.uni-hannover.de/pages/en:projekte_re_secreq.
[7] The dataset is available online at https://terapromise.csc.ncsu.edu/!/#repo/view/head/requirements/nfr.

requirements across the 15 student projects is around 11%, which is much lower than the realistic industry projects. This is reasonable as security is a practical concern that might not draw enough attention from students who have little working experiences. This dataset was used by Cleland-Huang et al. (2006) as a whole to train non-functional requirements classifiers, which include security as a particular type of non-functional requirements. We here make an assumption that students who specify more security requirements are more experienced in security requirements analysis. Thus, we divide the 15 projects into two groups based on the percentage of security requirements contained in the projects. In particular, projects that contain less than 10% security requirements (i.e., project 1, 2, 6, 7, 9, 10) are defined as the junior security group, while others are deemed as the senior security group (i.e., project 3, 4, 5, 8, 11, 12, 13, 14, 15). The purpose of having such a division is to investigate the correlation between the performance of our approach and the security expertise of the requirements analysts.

5.2. Research questions

RQ1. Which machine learning algorithm works best for our approach?

RQ2. To what extent classifiers trained by our approach can classify security requirements? Does our approach perform better than existing approaches?

RQ3. To what extent classifiers trained by our approach can be generalized to classify security requirements in other domains? Is our approach better than existing approaches?

RQ4. Does the quality of security requirements specification affect the performance of our approach?

5.3. Experiment design

To address the above research questions, we design four experiments accordingly.

Experiment 1. There are many off-the-shelf machine learning algorithms, which have been categorized into specific types. In this experiment, we focus on three particular types of algorithms that have been widely adopted, i.e., Naive Bayes (NB), Decision Tree (DT), Logistic Regression (LR). Specifically, Weka has corresponding implemented APIs, named NaiveBayes, J48, Logistic, respectively. A full list of algorithms that are provided by Weka can be found online[8]. In particular, we separately apply those algorithms to train classifiers based on each of the three requirements specifications in Knauss et al. (2011), and then compare their classification results.

Experiment 2. Based on the results of the first experiment, we adopt the best algorithm and further compare its performance with existing approaches. Given the three requirements specifications in Knauss et al. (2011), we prepare seven training datasets by enumerating all their combinations, except for the empty one. Then we apply our approach to each of these seven datasets and compare the classification results with existing approaches.

Experiment 3. In order to assess the generalizability of trained classifiers, we need to ensure the training data and test data belong to different domains, i.e., different requirements specifications. Specifically, we prepare six training datasets by traversing all combinations of the three requirements specifications in Knauss et al. (2011), except for the empty one and the union of the three specifications. Thus, classifiers trained from these six datasets will be applied to classify security requirements from other domains, i.e., the requirements specifications that are out of the training datasets. The classifications results will then be compared with existing approaches.

[8] http://www.cs.tufts.edu/~ablumer/weka/README.html#Classifiers.

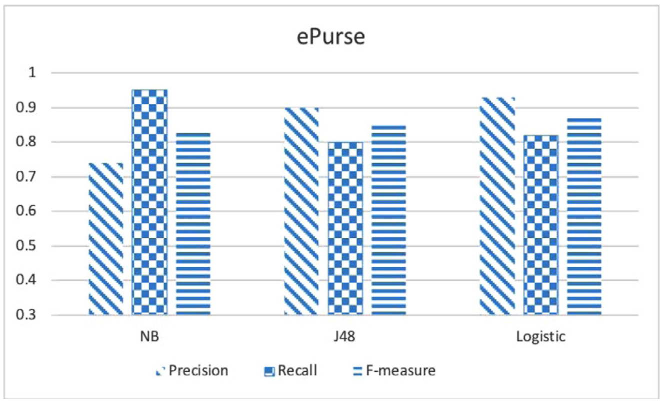


Fig. 7. Classification results of different algorithms on ePurse dataset

Experiment 4. We hypothesize that the security expertise of requirements analysts (i.e., people who are responsible for specifying security requirements) may affect the performance of our approach. To investigate this hypothesis, we train seven classifiers based on the enumeration of the three requirements specifications in Knauss et al. (2011) (except for the empty one), which are then applied to the junior group and the senior group in Cleland-Huang et al. (2006), respectively.

5.4. Evaluation metrics

We use standard metrics in the field of information retrieval, including *precision, recall*, and *f-measure* Baeza-Yates et al. (1999). In order to ensure the validity of our evaluation, we leverage k-fold cross validation Picard and Cook (1984) to avoid biases in the division of the dataset. For one thing, this technique can systematically create disjointed training data and test data based on a given dataset. For another, this technique can minimize the likelihood of data overfitting, which happens when classifiers are only trained based on particular training data.

When applying this technique, the dataset will be randomly divided into *k* parts with equal size, *k-1* parts of which are used for training and the remaining part is used for evaluating the trained classifier. This procedure will be iterated with different "*k-1* parts" each time. The choice of *k* is a trade-off. Choosing lower *k* makes the evaluation tasks less complicated and faster, but it implies less variance and may introduce more biases. Thus, we should experimentally determine *k* to ensure unbiased results while minimizing the complexity of our evaluation. According to Chantree et al. (2006) and Knauss et al. (2011), 10 is a decent value of *k* for this purpose. It is worth noting that cross validation cannot be applied in experiment 3 and (part of) experiment 4, because these experiments require the test data and training data strictly belong to different domains.

5.5. Results

Experiment 1. We separately adopt the aforementioned eight classification algorithms to train security requirements classifiers on each of the three requirements specifications. Figs. 7–9 present evaluation results on the three requirements specifications, respectively, which show significant variances among different domains. The average recall on the ePurse requirements specification (hereafter ePurse) is 0.86, while the average recall on CPN and GP are

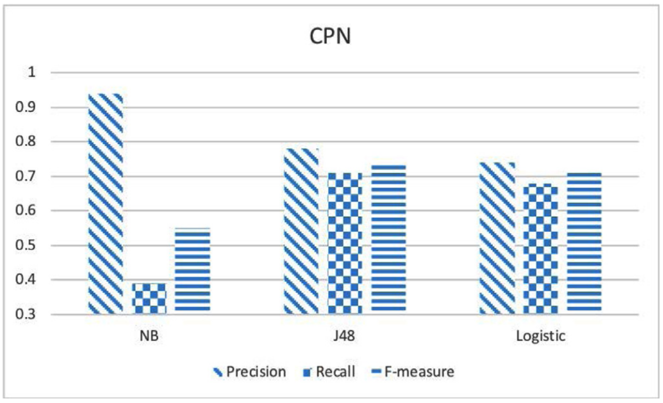


Fig. 8. Classification results of different algorithms on CPN dataset

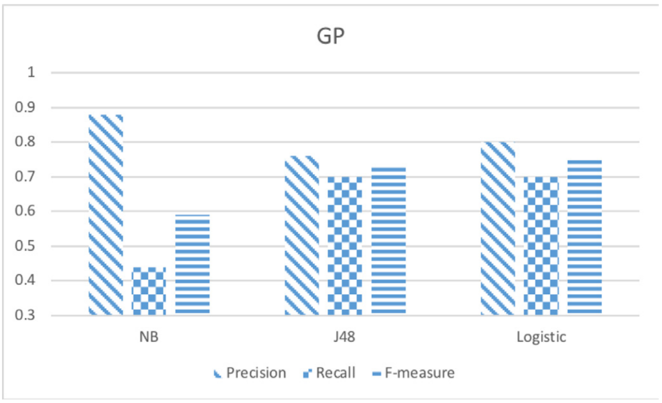


Fig. 9. Classification results of different algorithms on GP dataset

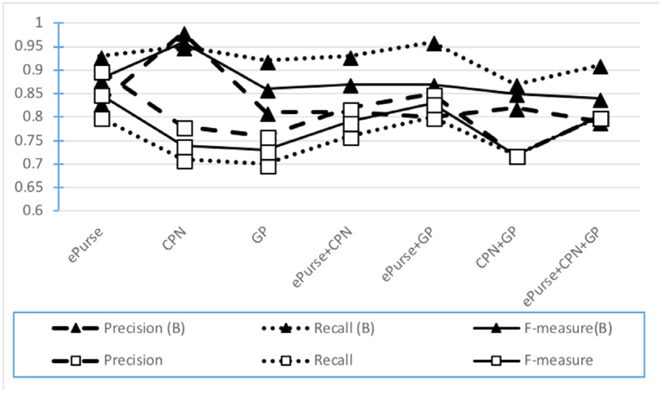


Fig. 10. Comparison on classifying requirements in same domains

only 0.59 and 0.61, respectively. Especially, the recall of NB has a sharp drop from 0.95 (ePurse) to 0.39 (CPN) and 0.44 (GP).

Overall, the Bayes-based algorithms (i.e., NB and BN) have the worst and most unstable performance. The tree-based algorithm (i.e., J48) and regression-based algorithm (i.e., Logistic) have similar performances. As emphasized by Berry et al., for CASE tools (or in general automated analysis tasks), recall is more important than precision (Berry et al., 2012). When classifying security requirements, the recall should be at least 0.7 Knauss et al. (2011). Among the eight classification algorithms, only J48 can reach at least 0.7 recall on all the three requirements specifications, and its average precision and recall are among the best.

Table 6
Compare our approach with existing studies

Dataset	Precision	Recall	F_1 score
ePurse	0.83/0.9	0.93/0.8	0.88/0.85
CPN	**0.98/0.78**	0.95/0.71	**0.96/0.74**
GP	0.81/0.76	**0.92/0.7**	0.86/0.73
ePurse+CPN	0.81/0.82	**0.93/0.76**	0.87/0.79
ePurse+GP	0.8/0.85	**0.96/0.8**	0.87/0.83
CPN+GP	0.82/0.72	**0.87/0.72**	0.85/0.72
ePurse+CPN+GP	0.79/0.8	0.91/0.8	0.84/0.8
average	0.83/0.8	**0.92/0.76**	0.88/0.78

Remark: The existing study results are presented first

Table 7
Comparison on classifying requirements in different domains

Dataset	Precision	Recall	F_1 score
ePurse(CPN)	**0.23/0.57**	0.54/0.56	**0.33/0.57**
ePurse(GP)	0.43/0.67	0.85/0.6	0.57/0.63
CPN(ePurse)	0.99/0.98	**0.33/0.49**	**0.47/0.66**
CPN(GP)	**0.29/0.85**	0.19/0.17	0.23/0.29
GP(ePurse)	**0.72/0.97**	**0.48/0.76**	**0.58/0.85**
GP(CPN)	**0.29/0.66**	0.65/0.76	**0.4/0.7**
ePurse+CPN(GP)	0.51/0.54	0.56/0.48	0.53/0.5
ePurse+GP(CPN)	**0.26/0.57**	0.85/0.71	**0.4/0.63**
CPN+GP(ePurse)	**0.84/0.98**	**0.31/0.73**	**0.46/0.84**
average	**0.51/0.75**	0.53/0.58	**0.44/0.63**

Remark: Results of the existing study are presented first

In particular, the average F_1 score of all classifiers trained by J48 can achieve 0.77, which has been slightly improved from our previous work in which the average F_1 is 0.76.

As such, we choose to use J48 in our approach for training security requirements classifiers.

Experiment 2. After selecting J48 as the default classification algorithm, we then evaluate the performance of our approach on various datasets, i.e., all combinations of the three requirements specifications, except for the empty one. We show the classification results of our approach in Table 6, and compare them with existing studies. Specifically, in each cell, we first present the results of the existing study (Knauss et al., 2011), following by which are our results.

Overall, our approach produces decent results in the sense that the precision and recall can reach at least 0.7 in all datasets. However, the compared approach performs better than ours. Especially, the average recall of the approach is 0.16 higher than ours, among which the CPN dataset reveals the biggest difference (as highlighted in Table 6). Such differences are more obvious as visualized in Fig. 10, where solid triangles indicate results of the existing study and hollow rectangles stands for our results.

When comparing the experimental results with our previous work Li (2017), both the average precision and recall have been slightly improved, and the average F_1 score increases by 0.01, from 0.76 to 0.77.

Experiment 3. In this experiment, we prepare training data using one or two of the three requirements specifications in Knauss et al. (2011), and test the trained classifiers using the remaining requirements specifications. Table 7 presents the evaluation results in the format *"results of the existing study/our results"*. Note that in the first column, the requirements specifications presented in parentheses are the test data.

On average, although it is not perfect, the f-measure of our approach can go beyond 0.6. Considering the difficulties in classifying requirements from completely different domains, we think the results are acceptable and our approach has shown the promising potential for dealing with such classification tasks. In particular,

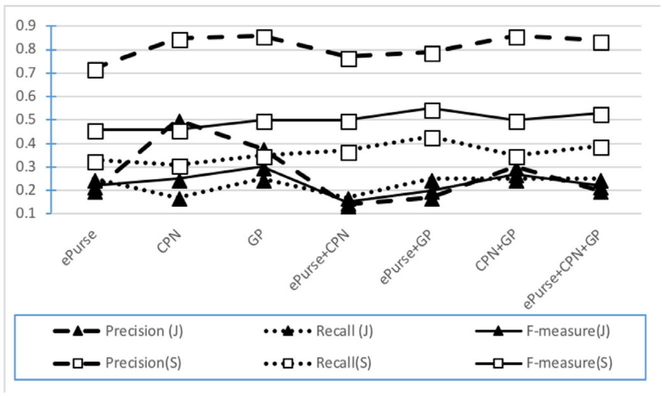


Fig. 11. Comparison on cross-domain classification on the junior group and the senior group

when comparing our results with existing studies, our approach performs significantly better. The average precision and recall of our approach are 0.24 and 0.05 higher than the existing study, respectively. Especially, in almost two-thirds of all cases (as highlighted in Table 7), the F_1 scores of our approach are at least 0.2 higher than the existing study.

When comparing the experiment results with our previous work (Li, 2017), the average precision has been improved while the average recall is not as good as before. Overall, the average F_1 score increases by 0.02, from 0.61 to 0.63.

Experiment 4. We use the seven classifiers we trained in Experiment 2 (as shown in Table 6) to classify requirements specified in the junior and senior groups, respectively. As visualized in Fig. 11, the evaluation results have shown a significant difference. In particular, the average precision of the senior group is far better than the junior group (0.81 vs 0.27), and the average recall of the senior group is also better (0.36 vs 0.23).

5.6. Summary and interpretation

We have answered our research questions via the three experiments. In general, both tree-based and regression-based classification algorithms can produce reasonable results, among which J48 has the best performance (**RQ1**). Due to the inherent complexity, security requirements can be incorrectly classified even by human, and there are inevitable noisy data in the datasets (discussed in the next section). Thus, one plausible explanation for the best performance of J48 is because J48 is comparatively more robust to noise. It is worth noting that this conclusion is aligned with our previous work, in which J48 also has the best performance over others, while our extended approach has reached a higher average F_1 score.

Although our approach can reach decent precision and recall (at least 0.7) in all datasets, the existing approach can do better than us (**RQ2**). On the other side, the classifiers created by our approach have shown promising potential to be generalized to classify other domains, which can achieve 0.75 precision and 0.58 recall on average. In this respect, our approach behaves significantly better than the existing approach (**RQ3**). For both of the research questions, our extended approach slightly outperforms our previous work in terms of F_1 score. The only decrease happens to the average recall of experiment 3, i.e., when the training and test data sets come from different domains. Overall, our extended approach proposed in this paper has slightly improved our previous work.

The differences between Knauss et al.'s work (Knauss et al., 2011) and our approach can be explained as follows. Their approach focuses on analyzing frequently occurred terms in security requirements, based on which they establish probability models for prediction. In other words, it actually mines potential documentation patterns of security requirements from a particular company, or in general, a particular application domain. Since each application domain may use domain-specific terms and have domain-specific ways of documenting security requirements, the existing approach can produce good classification results even if it does not consider the semantics of each requirements sentence. On the other hand, the existing approach is very likely to step into the problem of overfitting. As the frequent terms may significantly vary among different domains, the classifiers trained by the existing approach based on one particular application domain may have very poor classification results in another domain. Our approach does not delve into a particular application domain. Instead, we focus on the security semantics that should be expressed by security requirements. Specifically, we analyze such security semantics through security keywords and linguistic rules, which are actually domain-general. As a result, classifiers trained by our approach can produce reasonable results when applied to other application domains.

When using the classifiers that are trained based on the three industrial requirements specifications to classify requirements specified in student projects, we have perceived a significant difference between the junior security group and the senior security group. This confirms our hypothesis. Since our approach is established based on a formal ontology of security requirements, which comprehensively covers different definitions of security requirements, we actually combine domain knowledge of security requirements into the training of classifiers. As a result, when analysts specifying security requirements, if they have more security knowledge, they are more likely to correctly understand the definition of security requirements and specify them in an appropriate manner. Thus, we tentatively draw a conclusion that the security expertise of requirements analysts does have influences on the performance of our approach (**RQ4**).

6. Discussions

6.1. Challenges

A major challenge of our study is the reliability of the classified security requirements specifications. Different people, even security experts, can have different conceptual models of security requirements, and thus have different definitions of security requirements. Unfortunately, there is no unique definition that is widely accepted by academic researchers. For example, as summarized by Haley et al., security requirements can be represented as non-functional requirements, context, as well as constraints (Haley et al., 2008). As a result, when people classify security requirements, their results can be biased.

In our study, we use the security requirements specifications that have been manually classified by Knauss et al. The ePurse requirements specifications contain detailed information about the manual classification. In particular, five security experts are involved in the manual classification procedure, and the final classification results are subject to experts' votes. Two phenomena can be easily observed, illustrating the aforementioned problem. Firstly, there are many conflicts among security experts, i.e., sec *VS*. nonsec. Secondly, sometimes a security expert cannot really tell whether a requirement is a security requirement, i.e., unknown. As a result, many requirements cannot be clearly determined based on the vote. In particular, the ePurse specification initially contains 200 requirements, but only 124 can be classified for training. It is unclear how the other two requirements specifications are classified. If they are classified by another group of experts, the adopted classification criteria might be different, imposing a reliability threat to validity.

6.2. Aspects for improvement

Our proposed security keywords and linguistic rules have shown their usefulness in classifying security requirements, but they are by no means complete and should be further enhanced. Firstly, we can take into account more linguistic knowledge and explore more linguistic rules. For example, dependency analysis can reveal dependency relations among different terms, helping us to better analyze the semantics of security requirements. Secondly, as we have observed during the keywords mining, many security phrases are in the form of "adjective + noun", e.g., incorrect entry, incorrect action, incorrect configuration (Table 4). Since the noun can include a lot of candidates, it is less efficient to have each of such phrases as a single feature. Thus, we want to define several adjective phrases to optimize our classification results in the future. Lastly, in order to avoid the shortcoming of strict keyword matching, we plan to use *word2vec* to perform fuzzy matching based on the semantic distance between words.

7. Related work

7.1. Security requirements classification and identification

Given the importance of dealing with security requirements in early stages of system development, several approaches have been proposed to automatically classify security requirements from non-security requirements. The method proposed by Knauss et al. (2011), which has been compared in Section 5, is of the most relevance to our paper. In addition to the comparison we have presented throughout the paper, we further argue that these two approaches are applied in different contexts and serve different purposes, i.e., they can complement each other. In particular, if the to-be-classified requirements specifications are within the same application domains as the training requirements specifications, then Knauss's approach should be applied. On the contrary, if the to-be-classified requirements specifications come from a brand new application domain, our approach can contribute to this classification problem.

Munaiah et al. (2017) have similar intentions with us and have proposed a domain-independent classification approach using one-class classification models. Ideally, training a domain-independent security requirements classifier should be based on domain-independent specifications of security requirements. However, as the authors point out that there are no existing data sets containing domain-independent specifications of security requirements. Thus, they propose to collect training data from the solution domain (i.e., weaknesses and vulnerabilities) instead of the problem domain (i.e., requirements), and particularly focus on Common Weakness Enumeration (CWE) database. We acknowledge the challenge identified by the authors, and propose to tackle such a challenge by encapsulating the domain-independent security requirement knowledge into a security requirement ontology. Specifically, our approach also incorporates domain-independent security knowledge when identifying linguistic features of security requirements (i.e., Section 3.2). As for the evaluation, they have only shown the most effective classification results for each of the three test data sets, yielding a mean F_1 score of 67.68%. The similar metrics applied to our approach result in a mean F_1 score of 72.67%.

Wang et al. propose to identify security requirements in open source software projects by taking into account five project-related metrics, e.g., how many comments are made to a requirement (Wang et al., 2017). Such an approach introduces additional useful information that assists in training classifiers, which is essentially dealing with feature engineering. When corresponding information is available, we also believe it can render better classification results. On the contrary, our approach is built based only on the requirements specification, which applies to the situation when such project-related information is not available or not easy to obtain.

Another branch of study aims to automatically identify security-relevant sentences from requirements artifacts instead of classifying security requirements from other requirements (Riaz et al., 2014). After security-relevant sentences are identified, they define a list of context-specific security requirements templates to help translate such sentences into functional security requirements. This work complements ours in the sense that it applies to different objects. Our approach exclusively focuses on analyzing requirements specifications, which are supposed to be well documented and structured; while their approaches apply to natural language requirements artifacts in a more general sense, which might not be well structured.

7.2. Classification of non-functional requirements

Cleland-Huang et al. have proposed a probability-based approach for the automatic classification of various non-functional requirements (NFRs), including security requirements (Cleland-Huang et al., 2006; 2007). Their approach also mines frequent terms as indicators for classification. However, the more types of NFRs to classify, the more ambiguity and complexity are introduced. Overall, although the recall of their approach can reach 0.8, the precision is only around 0.21. In addition, their approach is applied to requirements specifications that are produced by master students in courses, which are less practical than the industry requirements specifications we used in this paper. It is worth mentioning that, as a precursor to this work, Cleland-Huang et al. have tried a keyword-based approach for classifying security and performance requirements. In particular, they extract a list of keywords from softgoal interdependency graphs (SIGs) (Chung et al., 2012). However, their approach does not consider security/performance conceptual models when extracting the keywords, and thus the keyword extraction procedure might not be systematic. Moreover, they have not taken into account any linguistic rules. Eventually, their approach can achieve 0.7 recall, but only 0.32 precision.

In addition to the above series of studies, other researchers deal with the classification of NFRs from different perspectives. Rashwan et al. (2013) propose to classify non-functional requirements based on ontology. However, the way they make use of ontology does not really unleash the power of ontology. In particular, when training NFR classifiers, they only refer to the categories defined in the ontology and do not leverage the semantics defined in each concept. In contrast, our proposal investigates the semantics of each security requirements concept and links it to corresponding linguistic features, directly contributing to the training of classifiers. Lu and Liang (2017) focus on classifying NFRs from massive user reviews of online application markets. As most user reviews are comparatively short, they propose to augment user reviews with similar words in the training set, which are then processed by the Bag-of-Words (BoW) model. In particular, they compare their proposal with a number of techniques, e.g., TF-IDF, BoW, Chi Squared, Naive Bayes and Decision Tree, the results of which show that augmented reviews can be well classified, and the BoW model can achieve the best results.

In order to help requirements engineering researchers to select appropriate machine learning based classification approaches, several recent studies have compared different approaches (Abad et al., 2017; Tóth and Vidács, 2018; Binkhonain and Zhao, 2019). In particular, Binkhonain and Zhao carried out a systematic review, covering 24 machine learning based classification approaches. They point three open challenges to NFRs classification: 1) lack of shared training data sets; 2) lack of standard definition; 3) feature identification and selection. Regarding such challenges, for one thing, our work deals with the standard definition issue by defining a formal ontology that clearly specifies definitions of each security requirements concept; for another thing, our approach identifies and select features of textual security requirements specification by systematically mapping ontological concepts to their linguistic representations.

7.3. Ontology-based security requirements analysis

Ontology is well known for being used to formalize domain knowledge. A number of existing studies base their security requirements analysis on security ontologies. Velasco et al. propose an ontological representation for defining reusable requirements, which can detect incompleteness and inconsistency in requirements (Velasco et al., 2009). In addition, their proposal can also contribute to specifying security requirements. Souag et al. propose a comprehensive security ontology for security requirements engineering based on 20 surveyed security ontologies (Souag et al., 2015). Thanks to such an ontology, they are able to reuse security knowledge which can answer informal and formal security requirements related questions, guiding security requirements analysis. Similarly, Salini and Kanmani have proposed a framework, which helps requirements engineers to identify assets, threats, vulnerabilities, and eventually contributes to the identification of security requirements (Salini and Kanmani, 2016). Our work aligns with the above studies in the sense that we also use ontology to encapsulate security requirements knowledge, which we argue is important in training security requirements classifiers.

7.4. Text-based requirements mining

There are a large number of studies on mining requirements from text. To name a few, Kaiya and Saeki propose to use domain ontology as domain knowledge for requirements elicitation, which is similar to our approach (Kaiya and Saeki, 2006). However, their approach relies on formal ontology and does not incorporate machine learning techniques. As such, their approach requires significant efforts to establish formal ontologies. Moreover, although their approach can reach almost 1 precision, their recall is only around 0.2. Morales-Ramirez et al. propose to mine Open-Source Software (OSS) forums in order to identify discussants intentions, i.e., “raw requirements” (Morales-Ramirez et al., 2014). They adopt similar techniques as ours, i.e., combining linguistic rules (speech act theory) with machine learning techniques. However, the performance of their approach is not as good as ours, and they have not evaluated their approach across different application domains. Massey et al. propose to automatically mining requirements from policy documents using Latent Dirichlet Allocation (LDA) (Massey et al., 2013). Their approach shows that the established topic models can indicate whether a document contains software requirements expressed as privacy protections or vulnerabilities. Such topic modeling techniques can be ideal complements to our linguistic-based analysis.

Saeki et al. have created security ontology for an application domain (SOAD), which contains security concepts of higher qualities (Saeki et al., 2013). In particular, their ontology covers security concepts, such as assets, threats, objectives, and security functional requirements, which can help us to further improve the set of security keywords.

8. Conclusions and Future Work

In this paper, we propose an ontology-based approach to automatically identify security requirements. In particular, we first defined an extended security ontology based on a comprehensive survey, which allows us to relate security requirements ontology at the conceptual layer to syntactic and lexical features of textual security requirements at the linguistic layer. In particular, we define a set of linguistic rules and security keywords that are normally used to describe security requirements, which are then used to train security requirements classifiers using typical machine learning algorithms. We have implemented a prototype to automatically extract linguistic features and train classifiers, with the help of which we have systematically carried out a series of experiments to evaluate our approach. The evaluation results show that our approach has promising potential to train classifiers that can be generalized to classify requirements from different application domains. In this respect, our approach performs far beyond the existing approaches. In addition, our another evaluation shows that the security expertise of requirements analysts has influences on our approach.

As for future work, we plan to incorporate further linguistic features into our approach in order for better classification results. In addition, we aim to incorporate cutting-edge NLP techniques (e.g., word2vec) into our approach to enhance the strength of linguistic analysis. Lastly, we are seeking to further evaluate our approach on more realistic requirements specifications, and ideally to support industry practices in reality.

Acknowledgements

This work is supported by National Key R&D Program of China (No. 2018YFB0804703) and the National Natural Science of Foundation of China (No.61902010).

References

Abad, Z.S.H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., Schneider, K., 2017. What works better? A study of classifying requirements. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). IEEE, pp. 496–501.
Baader, F., 2003. The description logic handbook: theory, implementation and applications. Cambridge university press.
Baeza-Yates, R., Ribeiro-Neto, B., et al., 1999. Modern Information Retrieval, 463. ACM Press New York.
Berry, D., Gacitua, R., Sawyer, P., Tjong, S., 2012. The case for dumb requirements engineering tools. Requir. Eng. 211–217.
Binkhonain, M., Zhao, L., 2019. A review of machine learning algorithms for identification and classification of non-functional requirements. Expert Syst. Appl.
Chantree, F., Nuseibeh, B., De Roeck, A., Willis, A., 2006. Identifying nocuous ambiguities in natural language requirements. In: Requirements Engineering, 14th IEEE International Conference, pp. 59–68.
Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J., 2012. Non-Functional Requirements in Software Engineering. Springer Science & Business Media.
Cleland-Huang, J., Settimi, R., Zou, X., Solc, P., 2006. The detection and classification of non-functional requirements with application to early aspects. In: Requirements Engineering, 14th IEEE International Conference. IEEE, pp. 39–48.
Cleland-Huang, J., Settimi, R., Zou, X., Solc, P., 2007. Automated classification of non-functional requirements. Requir. Eng. 12 (2), 103–120.
Committee, B., 2005a. BSI: IT-Grundschutz Catalogues. für Informationstechnik, Bundesamt.
Committee, I., 2005. Iso/iec 27002, information technology - security techniques - code of practice for information security management.
Committee, I., 2012. Iso/iec 27000:2012 information technology – security techniques – information security management systems – overview and vocabulary. Online: http://www.27000.org/.
Elahi, G., Yu, E., Zannone, N., 2009. A modeling ontology for integrating vulnerabilities into security requirements conceptual foundations. In: Conceptual Modeling-ER 2009. Springer, pp. 99–114.
Fenz, S., Ekelhart, A., 2009. Formalizing information security knowledge. In: Proceedings of the 4th international Symposium on information, Computer, and Communications Security. ACM, pp. 183–194.
Firesmith, D., 2003. Engineering security requirements. J. Obj. Technol. 2 (1), 53–68.
Firesmith, D., 2004. Specifying reusable security requirements. J. Obj. Technol. 3 (1), 61–75.
Firesmith, D.G., 2005. A taxonomy of security-related requirements. In: International Workshop on High Assurance Systems (RHAS'05). Citeseer, pp. 29–30.

Haley, C.B., Laney, R., Moffett, J.D., Nuseibeh, B., 2008. Security requirements engineering: a framework for representation and analysis. Softw. Eng. IEEE Trans. on 34 (1), 133–153.
Hoo, K.S., Sudbury, A.W., Jaquith, A.R., 2001. Tangible RoI through secure software engineering. Secure Bus. Q. 1 (2), Q4.
Institute, P., 2018. 2018 cost of data breach study: Global analysis.
Kaiya, H., Saeki, M., 2006. Using domain ontology as domain knowledge for requirements elicitation. In: Requirements Engineering, 14th IEEE International Conference. IEEE, pp. 189–198.
Kim, A., Luo, J., Kang, M., 2005. Security ontology for annotating resources. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, pp. 1483–1499.
Knauss, E., Houmb, S., Schneider, K., Islam, S., Jürjens, J., 2011. Supporting requirements engineers in recognising security issues. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, pp. 4–18.
Li, T., 2017. Identifying security requirements based on linguistic analysis and machine learning. In: Asia-Pacific Software Engineering Conference (APSEC), 2017 24th. IEEE, pp. 388–397.
Lu, M., Liang, P., 2017. Automatic classification of non-functional requirements from augmented app user reviews. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. ACM, pp. 344–353.
Massacci, F., Mylopoulos, J., Paci, F., Tun, T.T., Yu, Y., 2011. An extended ontology for security requirements. In: Advanced Information Systems Engineering Workshops. Springer, pp. 622–636.
Massey, A.K., Eisenstein, J., Antón, A.I., Swire, P.P., 2013. Automated text mining for requirements analysis of policy documents. In: Requirements Engineering Conference (RE), 2013 21st IEEE International. IEEE, pp. 4–13.
Miller, G.A., 1995. Wordnet: a lexical database for english. Commun. ACM 38 (11), 39–41.
Morales-Ramirez, I., Perini, A., Ceccato, M., 2014. Towards supporting the analysis of online discussions in OSS communities: a speech-act based approach. In: Forum at the Conference on Advanced Information Systems Engineering (CAiSE). Springer, pp. 215–232.
Mouratidis, H., Giorgini, P., Manson, G., 2003. An ontology for modelling security: the tropos approach. In: Knowledge-Based Intelligent Information and Engineering Systems. Springer, pp. 1387–1394.
Munaiah, N., Meneely, A., Murukannaiah, P.K., 2017. A domain-independent model for identifying security requirements. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). IEEE, pp. 506–511.
NIST, S., 2003. 800-53. Recommend. Secur. Control. Federal Inf. Syst. 800–853.
Picard, R.R., Cook, R.D., 1984. Cross-validation of regression models. J. Am. Stat. Assoc. 79 (387), 575–583.
Rashwan, A., Ormandjieva, O., Witte, R., 2013. Ontology-based classification of non-functional requirements in software specifications: a new corpus and svm-based classifier. In: 2013 IEEE 37th Annual Computer Software and Applications Conference. IEEE, pp. 381–386.
Riaz, M., King, J., Slankas, J., Williams, L., 2014. Hidden in plain sight: automatically identifying security requirements from natural language artifacts. In: 2014 IEEE 22nd International Requirements Engineering Conference (RE). IEEE, pp. 183–192.
Rudolph, M., Feth, D., Doerr, J., Spilker, J., 2016. Requirements elicitation and derivation of security policy templatesâan industrial case study. In: Requirements Engineering Conference (RE), 2016 IEEE 24th International. IEEE, pp. 283–292.
Saeki, M., Hayashi, S., Kaiya, H., 2013. Enhancing goal-oriented security requirements analysis using common criteria-based knowledge. Int. J. Softw. Eng. Know. Eng. 23 (05), 695–720.
Salini, P., Kanmani, S., 2016. A novel method: ontology-based security requirements engineering framework. In: 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS). IEEE, pp. 1–5.
Souag, A., Salinesi, C., Comyn-Wattiau, I., 2012. Ontologies for security requirements: a literature survey and classification. In: International Conference on Advanced Information Systems Engineering. Springer, pp. 61–69.
Souag, A., Salinesi, C., Mazo, R., Comyn-Wattiau, I., 2015. A security ontology for security requirements elicitation. In: International symposium on engineering secure software and systems. Springer, pp. 157–177.
Tóth, L., Vidács, L., 2018. Study of various classifiers for identification and classification of non-functional requirements. In: International Conference on Computational Science and Its Applications. Springer, pp. 492–503.
Tsoumas, B., Gritzalis, D., 2006. Towards an ontology-based security management. In: Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on, 1. IEEE, pp. 985–992.
Tulloch, M., 2003. Microsoft Encyclopedia of Security. Microsoft Press,.
Velasco, J.L., Valencia-García, R., Fernández-Breis, J.T., Toval, A., et al., 2009. Modelling reusable security requirements based on an ontology framework. J. Res. Pract. Inf. Technol. 41 (2), 119.
Wang, W., Hussein, N., Gupta, A., Wang, Y., 2017. A regression model based approach for identifying security requirements in open source software development. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). IEEE, pp. 443–446.

Tong Li holds a lecturer position in the Faculty of Information Technology at the Beijing University of Technology, China. His research interests are in the areas of software engineering, security requirements engineering, conceptual modeling and data mining. He is currently focusing on analyzing security requirements for social engineering attacks. He is now hosting a National Natural Science Foundation of China, a subtask of a National Key Research and Development Program of China, and a Beijing Education Science Planning Funding. He has been an author or co-author of more than 50 papers in peer-reviewed journals, conferences, or workshops. He is an expert of ISO/IEC JTC 1/ SC 27/ WG 4 and is serving as a co-editor of ISO 24392.

Zhishuai Chen is a bachelor student at Beijing University of Technology. Her research interests include machine learning and natural language processing. She is mainly engaged in applying natural language processing techniques to analyze security requirements.