Discovering Security Requirements from Natural Language Project Artifacts

John Slankas, Maria Riaz, Jason King, and Laurie Williams
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
+1(919) 515-7926
[john.slankas, mriaz, jtking, laurie_williams]@ncsu.edu

ABSTRACT

Project documentation often contains security-relevant statements that are indicative of the security requirements of a system. However these statements may not be explicitly specified or straightforward to locate. At best, requirements analysts manually extract applicable security requirements from project documents. However, security requirements that are not explicitly stated may not be considered during implementation. *The goal of this research is to aid requirements analysts in generating security requirements through identifying security-relevant statements in project documentation and providing context-specific templates to generate security requirements.* First, we identify the most prevalent security objectives from software security literature. To identify security-relevant statements in project documentation, we propose a tool-based process to classify statements as related to zero or more security objectives. We then develop a set of context-specific templates to help translate the security objectives of each statement into explicit sets of security functional requirements. We evaluate our process on six documents from the electronic healthcare software industry, identifying 46% of statements as implicitly or explicitly related to security. Our classification approach identified security objectives with a precision of .82 and recall of .79. From our total set of classified statements, we extracted 16 context-specific templates that identify 41 reusable security requirements.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Documentation, Reliability, Security, Standardization, Legal Aspects.

Keywords

Security, requirements, objectives, templates, access control, auditing, classification, constraints, natural language parsing.

1. INTRODUCTION

Developing a secure software-intensive system is a holistic process that requires an emphasis on security during all phases of software development lifecycle. There is often a lack of focus on security during early stages of software development that can lead to inadequately understood and improperly specified security requirements [24]. Understanding a system's underlying security objectives, such as confidentiality or accountability, can help focus the requirements engineering effort and facilitate development of an appropriate and relevant set of security requirements for the system under development.

Security objectives [14] are the security goals of a software system that can be operationalized by specifying an appropriate set of security functional requirements. Project documentation (such as functional requirements specifications, feature requests, bug reports, or applicable standards and certifications) often contains information related to the security objectives of software systems. However, these objectives may not be explicitly specified or straightforward to locate within the documentation. In prior research [33], we found that all of the examined documentation in the study contained security-relevant statements indicative of the security objectives and security requirements of the system. However, in many cases, these statements composed only a small percentage (under five percent) of number of lines in the document. Currently, experienced analysts must manually filter through the documentation to identify these security-relevant statements to obtain a more complete set of security requirements.

The goal of this research is to aid requirements analysts in generating security requirements through identifying security-relevant statements in project documentation and providing context-specific template to generate security requirements.

In this research, we propose a tool-based process to identify applicable security objectives (both explicitly-defined and implicitly-defined) for each statement in available project documentation. Based on patterns in the statements we classified, we further propose a set of 16 context-specific templates to generate security functional requirements.

First, we identified the most prevalent security objectives from software security literature. Our tool-based process parses existing documentation and applies machine learning algorithms to automatically classify statements with appropriate security objectives. The tool then provides a context-specific template for the requirement analysts to generate security requirements.

To evaluate our process, we analyzed six documents related to the development and maintenance of software systems in healthcare domain from the United States and Canada. We developed a labeled set of annotated statements by manually classifying statements in each of the six documents as relating to zero or more security objectives. For example, we annotate, "The system shall provide the ability to update and display a patient-specific medication list" as relating to confidentiality.

We also annotate the statement as relating to accountability since the system needs to keep a record of who performed the updates to the medication list. From these objectives, the tool provides templates associated with access control and logging for the requirement analyst to compose specific security requirements.

We use the following research questions to guide our analysis:

RQ1: From software security literature, what are the most prevalent security objectives of software systems?

RQ2: For the examined documentation, are there groups/sets of security objectives that appear consistently together?

RQ3: What features/elements do sentences of the same security objectives have in common?

RQ4: How effectively can security objectives be identified and extracted from selected set of documents?

Our research contributes the following:

- A repeatable process (and tool) to identify security objectives and to generate security requirements from project documentation
- A distribution report containing the frequency of identified security objectives in six healthcare documents
- A set of context-specific templates to help translate security objectives into functional security requirements.

The rest of this paper is organized as follows: Section 2 reviews the background for this paper. We discuss related work in Section 3. Section 4 presents the security objectives to address RQ1 followed by our proposed tool-based approach, Security Discoverer, in Section 5. In Section 6 we describe our research methodology. Section 7 presents results and evaluation to address RQ2-RQ4. We present the set of context-specific templates based on our analysis in Section 8. Section 9 discusses threats to validity for the study. Finally, Section 10 concludes the paper in addition to outlining future directions.

2. BACKGROUND

In this section we provide background information regarding security objectives and security requirements, machine classification and classification evaluation.

2.1 Security Objectives and Requirements

Security objectives are the security goals or desired security properties of a system [30]. Security requirements are functional and non-functional requirements that formalize security objectives without specifying how to achieve those objectives. Security functional requirements describe the desired security behavior of a system [6] and if incorporated, can achieve the corresponding security objectives. Security functional requirements can also be thought of as constraints on the functional requirements of the system [25] as they are meant to securely achieve a defined functionality of the system. For this paper, we use the term security requirements to mean security functional requirements.

2.2 Machine Learning and Classification

To identify security-related requirements from unconstrained natural language texts, we need flexible, yet effective, classification methods to handle different documents and multiple ways of expressing similar concepts. Machine learning provides such a foundation for our work. While techniques and algorithms vary widely in machine learning, they can be separated into two primary categories: supervised learning and unsupervised learning. In supervised learning, people train classifiers with labeled data. People and systems then use these classifiers to decide in which classification a previously unlabeled instance belongs. To be useful, a pre-trained classifier for a similar domain should be utilized. In contrast, unsupervised learning algorithms search data for common patterns (clusters). The data is not directly labeled; instead groups of common instances are created.

For this work, we utilize a combination of classifiers: k-nearest neighbor classifier (k-NN), multi-nominal naïve Bayes (NB), and Support Vector Machines (SVM). k-NN classifiers work by classifying a test item based upon which items previously classified are closest to the current test item. The classifier finds the k nearest "neighbors" and returns a majority vote of those neighbors to classify the test item. k-NN classifiers perform extremely well when they contain similar items (based upon a distance function) to the item currently under test. However, if similar items do not exist, the classification results are effectively random guesses. k-NN classifiers also work well in an interactive fashion due to their ability to incrementally learn as new items are classified and report similar sentences.

A naïve Bayes classifier works by selecting a class with the highest probability from a set of trained data sets given a specific document. Fundamentally, it assumes that each feature of a class exists independently of other features. Despite the simplification, the approach performs effectively in real-world problems. Naïve Bayes classifiers typically require fewer trained instances than other classifiers. SVM classifiers work by finding the optimal separator between two classes. As with naïve Bayes, text is represented as a word vector [20]. In our work, we utilize a k-NN classifier as the primary classifier unless it does not locate any similar sentences. At that point, we use a majority vote of the three classifiers to produce the classification result.

2.3 Classification Evaluation

To compare the results, we use recall, precision, and the F_1 measure. To compute these values, we first need to categorize the classifier's predictions into three categories for each classification value. True positives (TP) are correct predictions. False positives (FP) are predictions in which the sentence of another classification is incorrectly classified as the one under evaluation. False negatives (FN) are predictions in which a sentence of the same classification under evaluation is incorrectly placed into another classification. From these values, precision (P) is the proportion of correctly predicted classifications against all predictions for the classification under test: $\mathrm{P} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$. Recall is the proportion of classifications found for the current classification under test: $R = TP/(TP + FN)$. The F_1 measure is the harmonic mean of precision and recall, giving an equal weight to both elements: $F_1 = 2 \times \frac{P \times R}{P+R}$. From a security requirement perspective, recall is more important than precision in that we want to extract all relevant security requirements from the available documents. However, precision cannot be ignored because producing large amounts of false positives can frustrate users.

3. RELATED WORK

In this section, we discuss related work in terms of requirement classifications, natural language processing and security requirements engineering.

3.1 Requirement Classification

While text classification, especially with regard to Term Frequency - Inverse Document Frequency (TF-IDF), has been studied for a relatively long period of time [28], non-functional requirement (NFR) classification first appeared in the literature in 2006 [10]. In their work, Cleland-Huang et al. applied TF-IDF with an additional parameter to specify the frequency of indicator terms for a NFR category as compared to the appearance of those terms in the requirement currently under test. Their work performed well with a 0.8129 recall, meaning that they successfully found 81% of the possible NFRs in the dataset. However, their precision was 0.1244 indicating a large number of false positives. While they intentionally choose to maximize recall, users would be frustrated with their process due to the large numbers of false positives to examine and discard. Other researchers [9,37] have used the same dataset as Cleland-Huang, but instead adopt naïve Bayes and SVM classifiers. Both experiments reported higher scores for precision than the original research. Our approach utilizes an ensemble [27] of classification algorithms to produce classifications.

3.2 Identifying Security Requirements

Mellado et. al., have conducted a systematic review of security requirements engineering [24] to summarize existing approaches and respective contributions. Fabian et. al., also provide a comparison of security requirements engineering methods [12]. Methods for eliciting and documenting security requirements include framework-methods such as the SQUARE method [23], which provides a Capability Maturity Model-like reference model for coordinating various technical activities and artifacts; misuse or abuse cases[31]; anti-goals [21]; and assurance arguments [15]. Our work supports the identification of security functional requirements by bringing security-relevant statements to the immediate attention of requirements analyst along with a set of context-specific templates that can be used to generate security requirements supporting identified objectives for each statement. Further use/misuse-cases and risk models can be generated based on the set of security requirements identified using our approach to supplement the requirement specification process.

Firesmith [13] argues security requirements can be reusable across multiple systems and has proposed the use of parameterized templates to model reusable security requirements. The final step in our process is similar to Firesmith's approach in that we generate security requirements from a set of context-specific templates.

An important focus area related to security requirements engineering has been on extracting security requirements from regulatory texts ([8,22]). Other researchers have explored using natural language to generate access control policies from natural language ([18,35]). The focus or our research is on extracting security requirements from existing functional requirements and requirements-like documents and we do not consider issues related to regulatory compliance or policy specification in this paper.

4. SECURITY OBJECTIVES OF SOFTWARE SYSTEMS

By identifying the security objectives expressed or implied by a particular sentence within a document, we gain an understanding of the underlying intent of the statement as well as possible controls and mechanisms to establish that intent. While certain sets of security objectives are widely known such as the “Confidentiality, Integrity, and Availability (CIA) Triad”, we need to ensure the completeness of our security objective set.

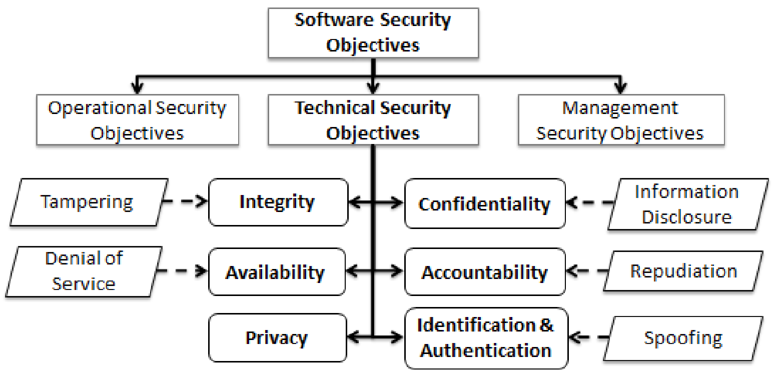
RQ1: *From software security literature, what are the most prevalent security objectives of software systems?*

To address this research question, we examined multiple security standards (National Institute of Standards and Technology Special Publication (NIST SP) 800-53 [7], NIST SP 800-33 [2], Federal Information Processing Standards (FIPS) publication 200 [5], FIPS publication 199 [4], Common Criteria [6], Federal Information Security, Management Act (FISMA) [3]), taxonomies of security objectives and requirements ([13,21]) and security seminal papers and books ([29,30]).

Software security is as much about physical protection of assets, training of personnel, planning and management as it is about building secure software. Security objectives of software systems therefore involve not only technical aspects from system development perspective but also operational and management aspects. For the purpose of this research however, we focus on technical security objectives of software systems. In Figure 1, we present a hierarchy of security objectives as identified from the literature. From the STRIDE Threat Model [19], we list corresponding attack classes that threaten a security objective where applicable.

We define each of the technical security objectives below. The references from where these objectives have been identified are listed after the objective's name. We also provide example statements from the set of documents we analyzed (see Section 6.1) that indicate the presence of the corresponding objective. The examples are numbered: <Document ID>-<Security Objective Abbreviation>.<#>.

Confidentiality (C) ([5-7,13,21,30]): The degree to which the "data is disclosed only as intended" [30]
CT-C.1: “The system shall provide the ability to update a patient



Software Security Objectives
Operational Security Objectives
Technical Security Objectives
Management Security Objectives
Tampering
Integrity
Confidentiality
Information Disclosure
Denial of Service
Availability
Accountability
Repudiation
Privacy
Identification & Authentication
Spoofing

Figure 1. Security Objectives Hierarchy

history by modifying, adding or removing items from the patient history as appropriate."

Integrity (I) ([5-7,13,21,30]): "The degree to which a system or component prevents unauthorized access to, or modification of, computer programs or data." [1]
ED-I.1: "When health information has been mistakenly associated with a patient, the system shall provide the ability to mark the information as erroneous in the record of the patient in which it was mistakenly associated and represent that information as erroneous in all outputs containing that information."

Identification & Authentication (IA) ([5-7,13,21]): The need to establish and verify the identity of a user, process or device.
ED-IA.1: "The system shall provide the ability to assign an identity to a patient at the time of arrival."

Availability (A) ([5-7,13,21,30]): "The degree to which a system or component is operational and accessible when required for use." [1]
ED-A.1: "It is essential that system response speed fast enough that there are no added delays in workflows in the ED when using the system."

Accountability (AY) ([5-7,13,21,30]): Degree to which actions affecting software assets "can be traced to the actor responsible for the action" [30]
ED-AY.1: "Every entry in the health record must be identified with the author and should not be made or signed by someone other than the author."

Privacy (PR) ([6,13,21]): The degree to which an actor can understand and control how their information is used.
NU-PR.2: "Nurses need to provide legitimate care in crisis situations that may go against prior patient consent directives ("break the glass" situations)"

5. SECURITY DISCOVERER

We now present our tool-based process, Security Discoverer (SD). The tool identifies any applicable security objectives for each sentence, provides context-specific templates to generate security requirements, and provides an authoring mechanism to finalize those requirements.

5.1 Overview

Figure 2 presents an overview of the four-step process and the associated tool. For input, the tool takes requirements-related natural language documents (requirement specifications, feature requests, etc.). Additionally, the organization needs a trained classifier for the current problem domain. This can be accomplished in one of three ways: 1) creating a new classifier by manually classifying sentences for security objectives from related projects; 2) utilizing an existing classifier; or 3) utilizing the tool in an interactive fashion to provide recommendations for classifications to aid the manual process. The tool has been designed to take classification corrections from the user and apply those corrections into the tool's current classifier. The tool parses the documents, identifies which (if any) security objectives relate to each statement within the document. The process then selects the relevant context-specific requirement template for each identified objective. The requirement analyst chooses either this template or another available template for the objective. The analyst then completes the appropriate security requirement from that chosen template.

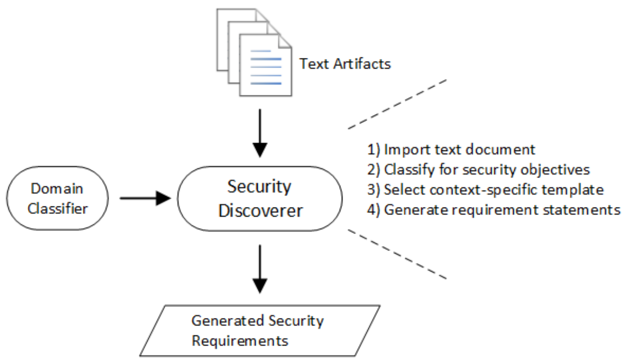

Text Artifacts
Domain Classifier
Security Discoverer
1) Import text document
2) Classify for security objectives
3) Select context-specific template
4) Generate requirement statements
Generated Security Requirements

Figure 2. Security Discoverer Process

5.2 Step 1: Import Text Document

The purpose of Step 1 is to import a text document into the SD tool such that each statement can then be classified to identify the security objectives. The tool first reads the entire text into the system. Next, to provide additional context and features for the classifier, the tool applies a concise document grammar (Figure 3) to label each statement from the text to a specific type:

- _title_: Statements which follow capitalization rules for titles. We separate these statements from other statements in our process as the titles rarely indicate a security-related requirement.
- _list start_: These statements represent the header or description of a list that follows.
- _list element_: These statements represent individual items contained within an ordered or unordered list. These statements are combined with the start of the list when sent to the parser and for classification. Combining the two provides additional context to both human analysts and machine classifiers.
- _normal sentence_: These statements are not considered as titles, list starts, or list elements.

Further, we identify heading and list identifiers (e.g., "4.1.1" and "•"). The process removes these identifiers from the statement passed into the Stanford Natural Language Parser (NLP). As list identifiers do not generally appear in articles on which the NLP parser was trained (such as news articles), the parser is not adequately trained to recognize such situations and produces results that are not consistent with what would otherwise be expected. If any irregularities are found during the parsing of the input text document, the parser defaults to "normal sentence" and continues processing text.

Within Figure 3, italicized words represent nonterminal symbols that can be replaced by other symbols on the right-hand side. Words in normal font are terminal symbols. Characters within quotation marks are also specific terminal symbols. λ represents an empty expansion of a nonterminal.

After identifying the different statement types, the tool parses each statement individually with the NLP and outputs a graph in the Stanford Type Dependency Representation (STDR) [11]. The tool then converts the STDR into SD's sentence representation (SR). The SR represents each statement as a directed graph where the vertices are words and the edges are the relationships between words. Figure 4 shows the SR for the statement, "The system shall automatically terminate a remote session after 30 minutes of inactivity." Although, in general the SR can be considered a tree, situations exist (primarily due to conjunctions) in which a vertex has multiple parents. Vertices correspond to words in the sentence and contain the word, the word's lemma, and the collapsed part of speech. Edges correspond to the relationship between two words (unchanged from the STDR). Using a pre-order traversal, the process creates the SR from the Stanford graph. As each vertex is created, we make two changes to the nodes. First, to avoid multiple versions of the same word, we use the lemma[1] of the original word. Second, to avoid differences in the part of speech, we collapse the parts of speeches for all nouns, verbs, and adjectives to their base category. For example, we treat all plural nouns and proper nouns as just nouns. Similarly, verbs with different tenses are treated collectively as a single group. We use a very small stop word list to remove common determiners[2] from the SR as demonstrated in Fig. 3 with the dashed lines. At this point, we have the text entered into the system and stored in a representation for the classifier.

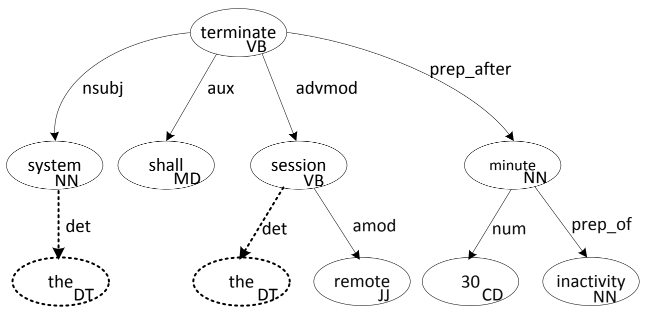


Figure 4. Sentence Representation

5.3 Step 2: Classify Sentences

Next, multiple machine learning algorithms are used to classify a statement into zero or more security objectives.

The process uses a k-NN classifier as the primary classifier. Such classifiers work by taking a majority vote of the existing classifications of the k nearest neighbors to the item under test. Thus, in our situation, to classify a statement, the classifier needs to find which existing classified statements are most similar to the current statement under test. k-NN classifiers use a distance metric to find the closest neighbors. This metric is the sum of the differences among the attributes used to determine the classification. Typically, Euclidean distance serves as a metric for numerical values while for nominal values (e.g., words), the distance is generally considered to be zero if both values are the same or one if they differ. Our situation is more complex as we have a variable number of attributes (words, parts of speech, named entities) to consider for each statement based upon the statement length. Additionally, certain words may be more closely related to one another than other words. As such, we utilize a custom distance metric based upon the distance between two SRs.

In prior work [32], we found that if we used a similarity threshold to determine whether or not to provide a classification answer, the k-NN classifier F_1 performance would be 1.0 (no misclassifications) , although not all of the statements would be classified. As such, we decided to utilize multiple machine learning algorithms to produce the final classification result. If the k-NN classifier's threshold is below a certain ratio (0.6) based upon the computed distance to the nearest neighbor(s) compared to the length of the sentence, we return the k-NN classifier's answer. Otherwise, we return a majority vote of the k-NN, naïve Bayes, and SVM. We term this classifier as "Combined SL."

For example, for the statement, "The system shall provide the ability to update a patient history by modifying, adding or removing items from the patient history as appropriate", the classifier should predict the following objectives: confidentiality, integrity, availability, and accountability.

Once the classification is complete, the user may review the

document	→	*line*
line	→	*listID* title *line* | title *line* | *sentence line* | λ
sentence	→	normalSentence | listStart (":" | "-") *listElement*
listElement	→	*listID sentence listElement* | λ
listID	→	*listParanID* | *listDotID* | number
listParanID	→	"(" *id* ")" *listParanID* | *id* ")" *listParanID* | λ
listDotID	→	*id* "." *listDotID* | λ
id	→	letter | romanNumeral | number

Figure 3. Document Grammar

predicted security objectives. If necessary, they can correct the classified objectives within the tool. Figure 5 shows a screenshot of the tool's user interface. The top table contains the document with individual columns to display the line number, statement type, assigned objectives and completion status, assigned cluster (groups of similar sentences, optional functionality), the statement themselves, and any generated requirements. The dialog in the lower left allows users to classify the statements for objectives. The area in the lower left allows the user to edit generated requirements from the selected templates (see Table 5 for example).

5.4 Step 3: Select Relevant Templates

Once the security objectives have been identified for a given statement, the user presses up button to bring up a list of context-specific templates for the security objectives and values (such as action or time) present within the statement. The user selects which templates to use. SD tracks which templates have been selected. The usage data provides the ability to determine which templates are most frequently used and in what combination. Additionally, the data could be used within a recommendation engine in future versions of the tool.

5.5 Step 4: Generate Requirement Statements

Once the requirement templates have been selected by the user, the tool presents the requirement text in an editor text window for the user to complete. In situations where a replaceable value has been found, the replacement is already made. For instance, if an availability statement specifies "during business hours", the tool detects the time period from the prepositional phrase and would automatically place that phrase into the generated requirement template. The tool maps generated requirements to source statements to produce a traceability matrix.

[1] A lemma is the base word form for a set of words. For instance, sang, sing, and sung all have the same lemma, "sing." Lemmas are more precise than stems as they take into account part of speech and other factors.

[2] a, an, the

6. RESEARCH METHODOLOGY

In this section, we discuss our methodology for collecting and preparing the selected documents for use within our study.

6.1 Study Documents

Security is an important consideration in a number of domains, including healthcare. For the purpose of this study, we have selected project documentation from healthcare systems, standards, and best practices primarily used in two different countries (United States and Canada) to increase the generalizability of our findings within the healthcare domain. To include a variety of document types in our study, we select the following six freely-available healthcare documents for our study:

- Certification Commission for Healthcare Information Technology (CCHIT)[3] Ambulatory Certification Criteria – a set of standards/certification criteria that outlines functional requirements of ambulatory EHR systems
- Nursing EHR and EHR Privacy and Security Requirements from the Canadian Health Infoway[4] -- two explicit sets of privacy and security requirements for Canadian EHR systems
- Emergency Department Information Systems Functional Document[5] – a set of functional data standards and conformance criteria provided by Health Level 7 International for EHR systems
- Open Source Clinical Application Resource (OSCAR) Feature Requests[6] – a set of informal descriptions of requests for additional functionality of the Canadian OSCAR EHR system, submitted by stakeholders (including users)
- Virtual Lifetime Electronic Records (VLER) user stories[7] – a set of functional requirements for the United States Department of Veteran's Affairs VLER technology initiative

Table 1 presents the complete list of documents along with a breakdown of statements classified for each security objective per document.

6.2 Domain Classifier

To develop a domain classifier for our tool, we first classify each statement in the six healthcare documents in terms of applicable security objectives. Three researchers manually read each natural language statement and classify the statement with relevant security objectives as follows:

1) Convert the document into text-only format.

2) Import one text document into SD Tool, and parse the document into individual statements using natural language processing.

3) Manually classify each statement in a document.

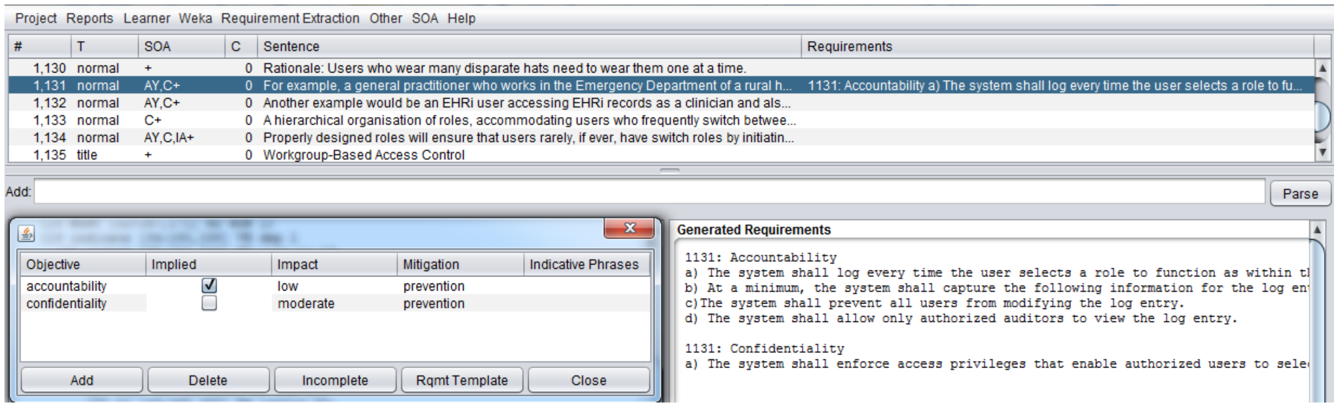
a) *Classification Phase*: For each document, two researchers individually classify each natural language statement to identify security objectives that apply to the statement. Each statement is classified in terms of zero or more security objectives. The classification phase results in the creation of two separate output files (one output file produced by each researcher) for each input document.

b) *Validation Phase*: A third researcher generates a difference report based on the two output files. The third researcher resolves the differences between the classifications of the first two researchers by communicating with the original two researchers to generate consensus for creating a final, consolidated classified corpus document.

The researchers spent a total of approximately 160 person-hours to create the domain classifier. Almost 46% of statements in the documents were identified as security-relevant. Given that we selected documents from industry standards and best practices related to healthcare domain, and that we also identified statements that *implied* a need for security, this number is significantly higher than our earlier findings [33]. We use the domain classifier for further analysis, including training the tool, to address our specific research questions.

6.3 Study Procedure

Once the domain classifier has been created, we executed four classifiers (the k-NN classifier, the "Combined SL" classifier, a multinomial naïve Bayes classifier and a SVM - sequential


Project Reports Learner Weka Requirement Extraction Other SOA Help
Add:
Parse
Add
Delete
Incomplete
Rqmt Template
Close

#	T	SOA	C	Sentence	Requirements
1,130	normal	+	0	Rationale: Users who wear many disparate hats need to wear them one at a time.	
1,131	normal	AY,C+	0	For example, a general practitioner who works in the Emergency Department of a rural h...	1131: Accountability a) The system shall log every time the user selects a role to fu...
1,132	normal	AY,C+	0	Another example would be an EHRi user accessing EHRi records as a clinician and als...	
1,133	normal	C+	0	A hierarchical organisation of roles, accommodating users who frequently switch betwee...	
1,134	normal	AY,C,IA+	0	Properly designed roles will ensure that users rarely, if ever, have switch roles by initiatin...	
1,135	title	+	0	Workgroup-Based Access Control	

Objective	Implied	Impact	Mitigation	Indicative Phrases
accountability	✓	low	prevention	
confidentiality		moderate	prevention	

Generated Requirements

```
1131: Accountability
a) The system shall log every time the user selects a role to function as within tl
b) At a minimum, the system shall capture the following information for the log en
c)The system shall prevent all users from modifying the log entry.
d) The system shall allow only authorized auditors to view the log entry.

1131: Confidentiality
a) The system shall enforce access privileges that enable authorized users to sele
```

Figure 5. Security Discoverer Tool Screenshot

[3] https://www.cchit.org/

[4] https://www.infoway-inforoute.ca/

[5] http://www.hl7.org/

[6] http://oscarcanada.org/

[7] http://www.va.gov/vler/

Table 1. Documents and Associated Security Objective Counts

			Security Objectives						
Doc. ID	**Document Title**	**# Lines**	**C**	**I**	**IA**	**A**	**AY**	**PR**	**None**
CT	CCHIT Certified 2011 Ambulatory EHR Criteria	331	252	214	19	14	260	5	7
ED	Emergency Department Information Systems Functional Document	2328	1162	1173	75	35	1354	76	773
NU	Pan-Canadian Nursing EHR Business and Functional Elements Supporting Clinical Practice	264	67	77	4	26	43	10	96
OR	OSCAR Feature Requests	5081	696	974	104	10	1184	18	3735
PS	Electronic Health Record (EHR) Privacy and Security Requirements	1623	146	120	43	31	149	85	928
VL	Virtual Lifetime Electronic Record User Stories	1336	693	731	13	19	797	10	375
	Total:	10963	3016	3289	258	135	3787	204	5914

minimal optimization classifier) on the document set. For each classifier considered, we tested using a stratified *n*-fold cross-validation and computed the precision, recall, and F_1 measure. With the *n*-fold cross-validation, data is randomly partitioned into *n* folds based upon each fold of approximately equal size and equal response classification. For each fold, the classifiers are trained on the remaining folds and then the contents of the fold are used to test the classifier. The *n* results are then averaged to produce a single result. We follow Han et al.'s recommendation [17] and use 10 folds as this produces relatively low bias and variance. The cross-validation ensures that all sentences are used for training and that each sentence is tested just once. In addition to our versions of the k-NN classifier, we utilized the multinomial naïve Bayes and SVM - sequential minimal optimization classifiers within Weka [16] suite. We directly utilized Weka classifiers through the available Java APIs. Since the Weka classifiers do not natively support multiple classifications for an item, we created individual classifiers for each algorithm and classification. As the folds are randomly generated, we executed the tests 3 times and averaged the results. To extract the top 20 keywords for each security objective, we utilized the information gain [26] attribute selector within Weka. Yang and Pedersen [36] found information gain to be the most effective method for feature selection in text classification.

7. EVALUATION

In this section, we address our research questions RQ2 - RQ4 as follows. We previouslyaddressed RQ1 in Section 4 in our discussion of identifying security objectives.

RQ2: Are there groups/sets of security objectives that appear consistently together?

Table 2 presents the 10 most frequently occurring security objective groups found within our set of 6 healthcare documents. Confidentiality and accountability each appear in 7 out of the 10 top objective groups, suggesting that confidentiality and accountability are common security objectives for healthcare systems. Integrity appears in 6 out of the 10 top objective groupings.

The confidentiality, integrity, and accountability security objectives appear *together* in the classifications of 2232 statements (20.4% of all statements classified), suggesting a strong relationship among the three. For example, the statement "The system shall provide a means to edit discharge instructions for a particular patient " [ED] implies that the confidentiality of discharge instructions should be maintained since it is protected health information; that the integrity of the discharge instruction data upon editing should be maintained; and that accountability should ensure that the user editing the discharge instructions can be held responsible.

Furthermore, confidentiality and accountability appear together in the classifications of 2859 statements (26.1% of all statements classified). Confidentiality and accountability are closely related security objectives. For example, the act of controlling access to sensitive data to help promote confidentiality is closely tied to the act of ensuring that a complete list of users who have accessed the sensitive data may be maintained for accountability. Therefore, in our domain classifier, statements that involve create/read/update/delete actions upon sensitive data are often classified as implying both confidentiality and accountability objectives.

Integrity and accountability appear together for 3119 statements (28.5% of all statements classified). Integrity involves ensuring that user interactions with sensitive data do not corrupt or somehow damage the state of the sensitive data. In terms of accountability, integrity helps ensure that the traces of user activity in the system may not be corrupted, modified, or damaged so that users can always be held accountable.

Privacy and identification/authentication objectives also appear in the top ten objective groupings, but are much less common. Privacy and identification/authentication often appear in combination with confidentiality, integrity, and/or accountability objectives. Overall, in the six healthcare documents in the study, the strongest relationships appear among confidentiality, integrity, and accountability.

Table 2. Frequently Occurring Objective Groups

Freq	Objective Group
2232	Confidentiality, Integrity, Accountability
702	Integrity, Accountability
443	Confidentiality, Accountability
106	Confidentiality, Integrity
104	Confidentiality, Identification & Authentication
98	Confidentiality, Accountability, Privacy
95	Integrity, Accountability, Privacy
90	Integrity, Identification & Authentication, Accountability
86	Confidentiality, Identification & Authentication, Accountability
83	Confidentiality, Integrity, Privacy

RQ3: What features/elements do sentences of the same security objectives have in common?

Table 3 presents the top twenty keywords listed for security objective. The set of keywords is very similar for confidentiality, integrity, and accountability objectives. This suggests a noticeable relationship among confidentiality, integrity, and accountability objectives.

For example, the keywords "system", "provide", and "ability" commonly appear in statements classified as confidentiality, integrity, and/or accountability. Statements classified as confidentiality, integrity, and/or accountability often appear in the form: "The system shall provide the ability to <action> <resource>". For example, "The system should provide the ability to check medications against a list of drugs noted to be ineffective for the patient in the past" [ED]. Since the resource in the example statement involves access to medications (protected health information), the statement is classified as implying a confidentiality objective. Likewise, since the statement involves access to protected information, the integrity of the data must be maintained. Finally, since the statement involves a user accessing protected information, the system should keep track of all users who have accessed the data so that they may be held accountable.

For identification/authentication, top keywords include, "authentication", "login", "username", "user", "authenticate", and "identify". While the structure of statements for confidentiality, integrity, and accountability share a common grammatical pattern, statements for identification/authentication share only common keywords that suggest the need to know the identity of a user, or the need to ensure that a user has authenticated into the system so that they can be identified by unique credentials.

Similarly, top keywords for availability include "run", "availability", "retain", "time", "destroy", "retention", and "real-time". Like identification/authentication, no grammatical pattern exists for availability statements. Instead, keywords that suggest temporal or data retention/destruction obligations are strong indicators of the presence of an availability security objective.

Top keywords for privacy include "consent", "phi", "disclosure", "purpose", and "privacy". Again, no grammatical pattern exists in the statements classified with the privacy security objective. Instead, common keywords that suggest privacy objectives include terms that involve a user (patients, in healthcare documents) choosing to give consent, or disclosure of protected information to anyone other than the patient. The disclose of protected information suggests that a user has consented to disclose the given information to a third-party.

Overall, keywords are the primary indicator of security objectives for identification/authentication, availability, and privacy. However, for many confidentiality, integrity, and accountability statements, the grammatical structure of the sentence is often the same. With similarities in grammatical structure and keywords within the statements of each security objective, we propose a set of context-specific templates for composing security requirements. We discuss the proposed templates in Section 8.

RQ4: How effectively can security objectives be identified and extracted from selected set of documents?

Table 4 presents the results of running the four classifiers against the six documents using a ten-fold cross validation. Creating the "Combined SL" demonstrated a slight performance gain over just using the Weka SMO classifier. The *k*-NN classifier performed equivalently to the SMO classifier. However, the advantage of *k*-NN classifier comes into play with using the SD tool in an interactive fashion. The classifier reports the sentences closest to the current sentence under test along with the distance. This allows an analyst to view similar sentences when making choices as to the possible security objectives. The reported precision of .82 implies that the tool correctly predicted 82% of those sentences it classified into a particular objective. The recall score of .79 means that it found 79% of all of the objectives possible. From an error perspective, the precision score implies that 18% of the identified objectives an analysts examines would be false positives and 21% of the

Table 3. Top 20 Keywords by Security Objective

Security Objective	Keywords
Confidentiality	system, provide, ability, patient, result, vler, exam, capture, datum, record, send, display, medication, information, list, requirement, status, consuming, order, complete
Integrity	system, provide, ability, vler, exam, send, capture, result, datum, store, consuming, patient, pass, click, pick-list, status, application, element, create, generate
Identification & Authentication	authentication, login, mac2002, username, oscar, user, authenticate, identify, cash, identity, myoscar, password, waitlist, log, registration, list2012, regen, uniquely, credentials, valid
Availability	run, availability, datum, retain, time, year, nurse, destroy, application, legally, recent, retention, care, maximum, real-time, information, period, destruction, record, historical
Accountability	system, ability, provide, vler, exam, result, send, consuming, click, pass, patient, capture, pick-list, datum, application, audit, status, store, record, list
Privacy	consent, patient, person, phi, disclosure, purpose, privacy, directive, require, organization, ehrus, law, authorization, information, connect, disclose, healthcare, inform, jurisdiction, collect

possible objectives were not found.

Table 4. Ten-Fold Cross Validation

Classifier	Precision	Recall	F_1 Measure
Naïve Bayes	.66	.76	.71
SMO	.81	.76	.78
k-NN (*k*=1)	.80	.76	.78
Combined SL	.82	.79	.80

8. CONTEXT-SPECIFIC TEMPLATES FOR SECURITY REQUIREMENTS

As a result of our analysis of the domain, we have developed a set of context-specific templates to generate security functional requirements based on the identified objectives from each statement. We have extracted 16 context-specific templates that identify 41 reusable security functional requirements. We list an example set of 5 of our context-specific templates, along with generated security requirements, in Table 5.

To use these templates, we first intend a requirements analyst to use the SD tool on the given project documentation. The tool then produces a set of security objective annotations for each statement in the documentation. Requirements analysts should consider our set of 16 context-specific templates[8] to determine which templates apply to each statement in the project documentation. For example, for a statement that the tool annotates as having a confidentiality objective, the requirements analyst should consider whether the confidentiality context-specific template C1 applies (see Table 5). If the statement contains a subject acting upon sensitive information, then the requirements analysis should compose a total of three security requirements to fulfill the one statement's confidentiality objective.

However, the newly composed security requirements also contain related security objectives, themselves. For example, the security requirement output in C1 for "The system shall enforce access privileges that <enable | prevent> <subject> to <action> <resource>" involves a user performing an action with sensitive information. This requirement suggests an accountability objective to track the user behavior defined in the requirement, so we include a reference from C1 to the related accountability-specific template AY1 for the requirements analyst to further consider. In Section 7, we discussed how security objectives for confidentiality, integrity, and accountability often appeared together in the classifications for 2232 statements. The cross-references in our context-specific templates for composing security requirements also reflects the strong relationships among confidentiality, integrity, and accountability.

9. THREATS TO VALIDITY

The following threats to validity exist for this research:

Selection of problem domain: We have evaluated our process in the domain of healthcare. The domain classifier created using documents from this domain may not be generalizable to other domains due to different security objectives and domain-specific vocabulary. Within the healthcare domain, however, we consider documents from the United States and Canada.

[8] A complete list of context-specific templates and labeled documents are available at: http://go.ncsu.edu/securitydiscoverer/

Selection of systems and documents: Security requirements may come from different types of documents / sources (policy documents, legislative texts, etc.) and variations may exist between security requirements of software systems, even in the same domain. Thus, selection of documents may influence the type and frequency of security-relevant statements that are identified.

Selection of security objectives: We have compiled a list of security objectives based on various taxonomies. Our list of security objectives may not be complete. To minimize this threat, we have looked at multiple sources from security literature so that we do not miss out important information from our list. A general consensus on the categorization of security objectives minimizes this threat.

Subjective assessment of security objectives: To develop the domain classifier, we carried out manual classification of statements which can be subjective. Misclassification of statements based on security objectives in the domain classifier may have occurred. To minimize this concern, two researchers have independently carried out the classification of each document while a third researcher has consolidated the final classification, lending subjectivity and validity to the process.

10. CONCLUSION

Our work describes a tool-based process for identifying key attributes of sentences to be used in security-related analysis and specification of security functional requirements using a set of context-specific templates. We have evaluated our process on six documents from the electronic healthcare software industry, identifying 46% of statements as implicitly or explicitly related to security. Our classification approach identified security objectives with a precision of .82 and recall of .79. From our total set of classified statements, we extracted 16 context-specific templates that identify 41 reusable security functional requirements. In addition to specifying security functional requirements, we provide a domain classifier of statements labeled with relevant security objectives. For practitioners, our research can additionally be used for gap identification in requirements specification.

We are considering the following directions for future work:

- *Access control*: How can the access control derived from natural language texts be appropriately modeled, checked for completeness and inconsistencies, and implemented in the environment?
- *Audit*: How can we evaluate the effectiveness of system's audit and non-repudiation capability?
- *Security requirement patterns*: From the identified security objectives, can we identify specific security requirement patterns [34] based on the objectives and context-specific templates (incorporating impact and mitigation approaches along with a discussion on requirements tradeoffs)?

11. ACKNOWLEDGEMENTS

This work is supported by the USA National Security Agency (NSA) Science of Security Lablet. Any opinions expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSA. We would like to thank the North Carolina State University Realsearch group for their helpful comments on the paper.

Table 5. Example Context-Specific Templates for Generating Security Requirements[9]

Objective: Confidentiality **Context:** *C1: Maintaining the confidentiality of data*	
Given: <subject> = user <resource> = sensitive information <action> = create/read/update/delete type actions ***Add Security Requirements:*** • The system shall enforce access privileges that <enable | prevent> <subject> to <action> <resource>. [see AY1, I1, I2, I4] • The system shall encrypt <resource> and store <resource> in encrypted format using an industry-approved encryption algorithm. [see AY3, I2, I4] • The system shall transmit <resource> data in encrypted format to and from the authorized <subject>. [see I4]	**Statement:** The system shall provide a means to edit discharge instructions for a particular patient. [ED] **Security Requirements:** • The system shall enforce access privileges that enable authorized users to edit discharge instructions for a particular patient. • The system shall encrypt discharge instructions and store discharge instructions in encrypted format using an industry-approved encryption algorithm. • The system shall transmit discharge instructions in encrypted format to and from authorized users.
Objective: Integrity **Context:** *I2: Maintaining integrity during write-type actions*	
Given: <subject> = system, user or role <resource> = sensitive information <action> = create / update / auto-populate / merge ***Add Security Requirements:*** • The system shall ensure that all mandatory information is provided for the <object> before <action>. • The system shall protect against loss of information during <action>. • The system shall have provision to report errors in < resource > after <action>. [see AY1] • The system shall have provision to correct errors in < resource > if errors are detected. [see AY1]	**Statement:** There must therefore be some capacity within the EHRi to merge multiple instances of patient records into a single record. [PS] **Security Requirements:** • The system shall ensure that all mandatory information is provided for the patient records before merging. • The system shall protect against loss of information during merging patient records. • The system shall have provision to report errors in patient records after merging. • The system shall have provision to correct errors in patient records if errors are detected.
Objective: Availability **Context:** *A1: Maintaining availability of data*	
Given: <time> = length of time, typically one year <keywords> = retention | archive | history <resource> = information <action> = read / view / display / send / receive / access ***Add Security Requirements:*** • The system shall store and make available <object> for a period of at least <time period>. [see C1] • The system shall provide the capability for an administrator to purge data that is at least <time period> old and in accordance with organizational retention policy. [see I4]	**Statement:** VLER DAS stores event descriptions in an audit log for a minimum of six (6) years. [VL] **Security Requirement:** • The system shall store and make available audit event descriptions for a period of at least 6 years. • The system shall provide the capability for an administrator to purge data that is at least 6 years old and in accordance with organizational retention policy.
Objective: Accountability **Context:** *AY1: Logging transactions with sensitive data*	
Given: <subject> = user or role <action> = create/read/update/delete <resource> = sensitive information ***Add Security Requirements:*** • The system shall log every time <subject> performs the <action> on <resource>. • The system shall allow only authorized auditors to view the log entry. [see I1, AY1]	**Statement:** The system should provide the ability to check medications against a list of drugs noted to be ineffective for the patient in the past [ED] **Security Requirements:** • The system shall log every time the user checks medications against a list of drugs noted to be ineffective for the patient in the past. • The system shall allow only authorized auditors to view log entry.
Objective: Privacy **Context:** *PR1: Usage of personal information*	
Given: <subject> = user or role <resource> = private or personally identifiable information <action> = create/read/update/delete/disclose/access ***Add Security Requirements:*** • The system shall inform the owner of <resource> of all the possible uses of <resource> that are authorized. [see C1] • The system shall allow the owner of <resource> to be notified when the <resource> is <action> by <subject>. [see I2, AY1] • The system shall have the ability to get consent from the owner of <resource> before accessing <resource> for authorized use. [see C1, AY1]	**Statement:** Nurses require access to historical patient data to support patient interaction and care planning [NU] **Security Requirements:** • The system shall inform the owner of historical patient data of all the possible uses of historical patient data that are authorized. • The system shall allow the owner of historical patient data to be notified when the historical patient data is accessed by nurses. • The system shall have the ability to get consent from the owner of historical patient data before accessing historical patient data for authorized use.

[9] A complete list of context-specific templates and labeled documents are available at: http://go.ncsu.edu/securitydiscoverer/

12. REFERENCES

[1] 1990. IEEE Standard Glossary of Software Engineering Terminology: http://standards.ieee.org/findstds/standard/610.12-1990.html

[2] 2001. Underlying Technical Models for Information Technology Security: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf

[3] 2002. Federal Information Security Management Act: http://csrc.nist.gov/drivers/documents/FISMA-final.pdf

[4] 2004. Standards for Security Categorization of Federal Information and Information Systems: http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf

[5] 2006. Minimum Security Requirements for Federal Information and Information Systems http://csrc.nist.gov/publications/fips/fips200/FIPS-200-final-march.pdf

[6] 2012. Common Criteria for Information Technology Security Evaluation,Version 3.1. Release 4: http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf

[7] 2013. Security and Privacy Controls for Federal Information Systems and Organizations: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

[8] Breaux, T.D. and Antón, A.I., 2008. Analyzing Regulatory Rules for Privacy and Security Requirements. *IEEE Transactions on Software Engineering 34*, 1 (Jan.), 5-20.

[9] Casamayor, A., Godoy, D., and Campo, M., 2010. Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology 52*, 436-445. DOI= http://dx.doi.org/10.1016/j.infsof.2009.10.010.

[10] Cleland-Huang, J., Settimi, R., and Solc, P., 2006. The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. In *14th IEEE International Requirements Engineering Conference (RE'06)* Ieee, 39-48. DOI= http://dx.doi.org/10.1109/RE.2006.65.

[11] De Marneffe, M.-C., Maccartney, B., and Manning, C., 2006. Generating Typed Dependency Parses from Phrase Structure Parses. *Proceedings of Language Resources and Evaluation*, 449-454. DOI= http://dx.doi.org/10.1.1.74.3875.

[12] Fabian, B., Gürses, S., Heisel, M., Santen, T., and Schmidt, H., 2010. A comparison of security requirements engineering methods. *Requirements Engineering - Special Issue on RE'09: Security Requirements Engineering 15*, 1, 7-40.

[13] Firesmith, D., 2004. Specifying Reusable Security Requirements. *Jornal of Object Technology 3*, 1 (Jan-Feb.), 15.

[14] Firesmith, D.G., 2003. Engineering Security Requirements. *Journal of Object Technology 2*(Jan-Feb), 16.

[15] Franqueira, V.N.L., Tun, T.T., Yu, Y., Wieringa, R., and Nuseibeh, B., 2011. Risk and argument: A risk-based argumentation method for practical security. In *Proceedings of the IEEE International Requirements Engineering Conference* (2011), 10.

[16] Hall, M., National, H., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H., 2009. The WEKA Data Mining Software : An Update. *SIGKDD Explorations 11*, 10-18. DOI= http://dx.doi.org/10.1145/1656274.1656278.

[17] Han, J., Kamber, M., and Pei, J., 2011. Data Mining: Concepts and Techniques, 744.

[18] He, Q. and Antón, A.I., 2009. Requirements-based Access Control Analysis and Policy Specification (ReCAPS). *Information and Software Technology 51*, 993-1009. DOI= http://dx.doi.org/10.1016/j.infsof.2008.11.005.

[19] 2006. Uncover Security Design Flaws Using The STRIDE Approach: http://msdn.microsoft.com/en-us/magazine/cc163519.aspx

[20] Joachims, T., 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*. DOI= http://dx.doi.org/10.1007/BFb0026683.

[21] Lamsweerde, A.V., 2004. Elaborating Security Requirements by Construction of Intentional Anti-Models. In *Proceedings of the International Conference on Software Engineering (ICSE 2004)* (Edinburgh, Scotland2004), 10.

[22] Maxwell, J.C., Antón, A.I., and Swire, P., 2011. A legal cross-references taxonomy for identifying conflicting software requirements. In *Requirements Engineering*, 197-206.

[23] Mead, N.R., Houg, E.D., and Stehney, T.R., 2005. *Security Quality Requirements Engineering (SQUARE) Methodology*. Software Engineering Inst.

[24] Mellado, D., Blanco, C., Sánchez, L.E., and Fernández-Medina, E., 2010. A systematic review of security requirements engineering. *Computer Standards & Interfaces 32*, 4, 153-165.

[25] Moffett, J.D., Haley, C.B., and Nuseibeh, B., 2004. *Core security requirements artifacts*. The Open University.

[26] Quinlan, J.R., 1986. Induction of decision trees. *Machine Learning 1*, 1.

[27] Rokach, L., 2010. Ensemble-based classifiers. *Artificial Intelligence Review 33*, 1-2, 1-39. DOI= http://dx.doi.org/http://dx.doi.org/10.1007%2Fs10462-009-9124-7.

[28] Salton, G. and Mcgill, M.J., 1986. Introduction to Modern Information Retrieval.

[29] Saltzer, J.H. and Schroeder, M.D., 1974. The Protection of Information in Computer Systems. *Communication of the ACM 17*, 7.

[30] Schumacher, M., Fernandez-Buglioni, E., Hyberston, D., Buschmann, F., and Sommerlad, P., 2006. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, Ltd, West Sussex.

[31] Sindre, G. and Opdahl, A.L., 2005. Eliciting Security Requirements with Misuse Cases. *Requirements Engineering 10*, 12. DOI= http://dx.doi.org/10.1007/s00766-004-0194-4.

[32] Slankas, J. and Williams, L., 2012. Classifying Natural Language Sentences for Policy. *2012 IEEE International Symposium on Policies for Distributed Systems and Networks*, 33-36. DOI= http://dx.doi.org/10.1109/POLICY.2012.16.

[33] Slankas, J. and Williams, L., 2013. Automated Extraction of Non-functional Requirements in Available Documentation. In *International Conference on Software Engineering (ICSE) 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*, 9-16.

[34] Withall, S., 2007. *Software Requirement Patterns.* O'Reillly.

[35] Xiao, X., Paradkar, A., Thummalapenta, S., and Xie, T., 2012. Automated Extraction of Security Policies from Natural-Language Software Documents. In *International Symposium on the Foundations of Software Engineering (FSE)*, Raleigh, North Carolina, USA.

[36] Yang, Y. and J.P., P., 1997. A Comparative Study on Feature Selection in Text Categorization In *Fourteenth International Conference on Machine Learning (ICML'97)*, 412-420.

[37] Zhang, W., Yang, Y., Wang, Q., and Shu, F., 2011. An Empirical Study on Classification of Non-Functional Requirements. In *The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, 190-195.